



Numéro d'ordre : 41289

Université Lille 1, Sciences et Technologies

THÈSE

présentée pour obtenir le titre de docteur
spécialité Informatique

par

AMEN SOUISSI

MODÉLISATION CENTRÉE SUR LES PROCESSUS MÉTIER POUR LA GÉNÉRATION COMPLÈTE DE PORTAILS COLLABORATIFS

Thèse soutenue le 20 décembre 2013, devant la commission d'examen formée de :

Pierre-Alain Muller	Professeur Université Haute-Alsace	Rapporteur
Marie-Pierre Gervais	Professeur Université Paris Ouest	Rapportrice
Pierre Boulet	Professeur Université de Lille 1	Directeur
Cédric Dumoulin	Maître de Conférence Université de Lille 1	Encadrant
Michael Launay	Dirigeant Ecréall	Invité

Université Lille 1, Sciences et Technologies
LIFL - UMR 8022 - Cité Scientifique, Bât. M3 - 59655 Villeneuve d'Ascq Cedex

MODÉLISATION CENTRÉE SUR LES PROCESSUS MÉTIER POUR LA GÉNÉRATION COMPLÈTE DE PORTAILS COLLABORATIFS

AMEN SOUISSI

Thèse de doctorat

20 décembre 2013

RÉSUMÉ

Les entreprises collaborent pour saisir des opportunités, échanger des documents et ressources, cela en suivant des processus métier pouvant évoluer. Les portails collaboratifs sont une solution orientée web à ce besoin de collaboration. Cependant, la conception et la maintenance d'un portail collaboratif métier n'est pas trivial et reste peu accessible aux acteurs de l'entreprise. Cela a comme conséquence la difficulté de maintenir et faire évoluer le portail collaboratif sans que cela ne soit trop coûteux en temps et financièrement. Afin de répondre à cette problématique, une solution consiste à capter les besoins métier de la collaboration dans un modèle, puis générer automatiquement le portail collaboratif correspondant. Le modèle, dans ce cas, doit être accessible aux acteurs métier et expressif décrivant ainsi les aspects les plus complexes d'une collaboration. C'est dans ce contexte que se situent nos travaux. À défaut d'avoir une solution toute faite, nous avons mis en place une approche de conception de portail collaboratif fondée sur l'Ingénierie Dirigée par les Modèles. Pour la description de nos portails, nous avons choisi de privilégier la modélisation des entreprises centrée sur les processus métier comme point de départ. Notre solution repose sur notre métamodèle MACoP (Modeling and Analysis of Collaborative Portal). Dans ce métamodèle nous avons fait cohabiter l'accessibilité et l'expressivité. Cela en proposant de nouveaux concepts permettant ainsi la génération complète des portails collaboratifs. Le métamodèle MACoP est accompagné d'une chaîne de transformations permettant de passer directement d'un modèle MACoP au code Python du portail collaboratif.

Mots clés : Portail collaboratif, Application web, Processus métier, Système d'information, Modélisation des entreprises, Ingénierie Dirigée par les Modèles (IDM), Transformation de modèles, Génération de code.

ABSTRACT

Companies collaborate to seize opportunities as well as exchange documents and other types of resources. This is achieved by following business processes that are subject to evolution. Collaborative portals are web oriented solutions aimed at this need of collaboration. However, the development and maintenance of a collaborative portal is non-trivial and remains hardly accessible for many companies. As a consequence, the challenge is controlling the costs of maintenance and implementing new features. To circumvent these issues, it is possible to collect business requirements of the collaboration in a model and then generate automatically the corresponding collaborative portal. The model, in this case, must be accessible by the business actors and must express even the most complex aspects of the collaboration needs. This is the context of our work. Having not found a solution which corresponded to our requirements, we have developed a design approach of collaborative portals founded on the Model Driven Engineering. As for the description of our portals, we have chosen to focus on the business modeling based on the business processes like starting point. Our solution relies on our meta-model MACoP (Modeling and Analysis of Collaborative Portal). In this meta-model we have joined together accessibility and the expressivity, by proposing new concepts allowing the complete generation of collaborative portals. The meta-model MACoP is accompanied by a transformations chain that makes it possible to pass directly from a MACoP model to the Python code of the collaborative portal.

Keywords : Collaborative portal, web application, business process, information system, business modeling, Model Driven Engineering (MDE), Models transformation, code generation.

REMERCIEMENTS

Au-delà de cette thèse, je tiens à exprimer ma profonde reconnaissance à mon directeur Pierre Boulet de m'avoir ouvert les portes de la recherche et de m'avoir soutenu pendant cette thèse. Mes profonds remerciements reviennent aussi à Cédric Dumoulin pour son encadrement, pour son écoute et pour ses conseils lors de l'écriture de ce manuscrit. De même, je remercie Michael Launey de m'avoir accueilli dans son équipe et pour sa confiance, mon carburant principal. Je témoigne également une profonde gratitude pour les rapporteurs de cette thèse, Marie-Pierre Gervais et Pierre-Alain Muller, d'avoir lu et commenté ce manuscrit, ainsi que pour leurs remarques et questions.

Je tiens à remercier, aussi, les membres de l'équipe Ecréall pour leur sympathie et pour l'ambiance amicale qui a gouverné le long de cette thèse. En particulier, je remercie Vincent Fretin pour le remarquable travail de développement dont il a fait preuve. Également, je remercie Cédric Méciant pour sa lecture de ce manuscrit, ainsi que pour ses remarques. Sans oublier les membres de l'équipe DaRT du LIFL pour leur accueil.

Enfin, je souhaiterais remercier ma famille et mes amis de m'avoir soutenu et aidé au cours de ma thèse. En particulier, je remercie mon fils Elias Souissi pour l'espoir qu'il me donne, mon épouse Elena Souissi pour sa patience et mes parents Amna et Abdelmajid Souissi pour leur soutien et leur prière qui ne cesse jamais.

TABLE DES MATIÈRES

Première partie

INTRODUCTION

1 INTRODUCTION

Sommaire

La collaboration est une nécessité pour l'entreprise, que ce soit en interne, entre ses collaborateurs, ou en externe, avec d'autres entreprises ou avec des clients. Une collaboration se concrétise généralement par le partage de documents, de ressources, de services ou de connaissances. Ce partage se fait en suivant des règles métiers indiquant par exemple quel collaborateur peut créer ou manipuler le document ou la ressource, à quel moment, et enfin quelles étapes doivent être suivies afin d'aboutir au produit final. L'informatique, et les technologies de l'internet permettent de répondre à ce besoin de collaboration, sous la forme de portail collaboratif. Cet outil informatique, généralement constitué d'un serveur web, réalise l'interface entre les différents systèmes d'information impliqués dans les collaborations. Ce portail permet à un acteurs métier donné d'interagir avec les informations d'une collaboration ainsi que d'interagir avec les autres acteurs en suivant des processus métier bien définis. La figure 1, illustre la position des portails collaboratifs par rapport aux applications web classiques centrées sur les données et les solutions de gestion des processus métier (BPM) permettant l'orchestration des processus métier. Dans le contexte de nos travaux, le portail collaboratif est considéré comme une application web permettant une exécution, centrée sur les données, des processus métier. Dans cette thèse, nous nous intéressons à la génération complète des portails collaboratifs à partir d'un modèle décrivant les exigences métier.

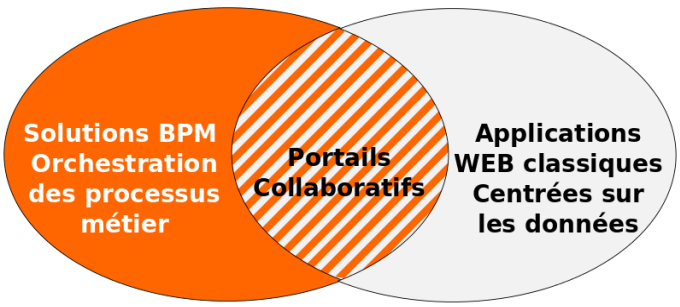


FIGURE 1: Le portail collaboratif comme une application web permettant une exécution, centrée sur les données, des processus métierr

Plusieurs approches permettent la modélisation et la génération des applications web suivant différentes démarches et pour différents domaines, de l'interface homme-machine (IHM) à l'orchestration des processus. Mais, aucune de ces approches ne permet de générer de portail collaboratif adapté au métier de l'entreprise à partir d'un modèle métier à haut niveau d'abstraction indépendamment de la technologie cible. En effet, actuellement, la modélisation des applications web est centrée, principalement, sur le concept de l'hypertexte et non sur les exigences métier. Ces dernières sont décrites d'une manière implicite et sont mises à plat ce qui rend le modèle volumineux et difficilement accessible par les acteurs métier. Les solutions BPM quant à elles ont favorisé la modélisation des exigences métier et négligé l'expressivité des modèles. De ce fait, les applications générées sont limitées exprimant ainsi une collaboration fédérée. Ces applications, se limitent à l'orchestration des processus métier et non à leur exécution naturelle (c'est-à-dire l'exécution centrée sur les données des processus métier en arrière-plan).

À l'étude de ces approches, nous nous sommes posé des questions principales comme « *Quels sont les critères à satisfaire pour avoir une modélisation ergonomiquement efficace par rapport aux acteurs métier ?* », « *Quels sont les concepts métier permettant de passer d'une modélisation centrée sur les hypertextes à une modélisation centrée sur les besoins métier ?* » ou, « *Comment avoir un niveau d'expressivité élevé ainsi qu'un niveau d'accessibilité élevé dans notre modélisation ?* ». Notre approche donne une réponse à ces questions et ouvre la voie vers une modélisation centrée sur les exigences métier des applications web.

Pour une entreprise, le portail collaboratif évolu suivant son système d'information et les nouvelles technologies. Cette évolution doit être rapide, efficace et à moindre coût. Toutefois ceci n'est possible que si l'entreprise est structurée, c'est-à-dire si elle a formalisé sa façon de travailler. Les formes de structuration les plus matures reposent sur la modélisation des entreprises, par exemple à l'aide de BPMN. Convaincus que l'entreprise doit avoir le plus d'autonomie possible sur l'évolution de son système d'information, étant donné qu'elle est la seule à le connaître précisément, nous avons mis en place une méthodologie de conception de portail collaboratif fondé sur l'Ingénierie Dirigée par les Modèles (IDM) [66]. Pour la description de nos portails collaboratifs, nous avons choisi de privilégier la modélisation des entreprises centrée sur les processus métier comme point de départ.

La figure 2 montre l'utilité d'une approche IDM pour la génération et l'évolution des portails collaboratifs selon les évolutions de la collaboration. Une approche IDM, permettant une génération complète des portails collaboratifs décrits à un haut niveau d'abstraction, permet aux acteurs métier de mettre à jour leurs modèles décrivant la collaboration à chaque évolution de celle-ci (ajout d'un nouveau col-

laborateur, ajout d'un nouveau type de document, etc.). La figure 2 montre, aussi, les différents domaines d'ingénierie traités dans cette thèse : l'ingénierie des systèmes d'information ; l'ingénierie dirigée par les modèles et enfin l'ingénierie des applications web.

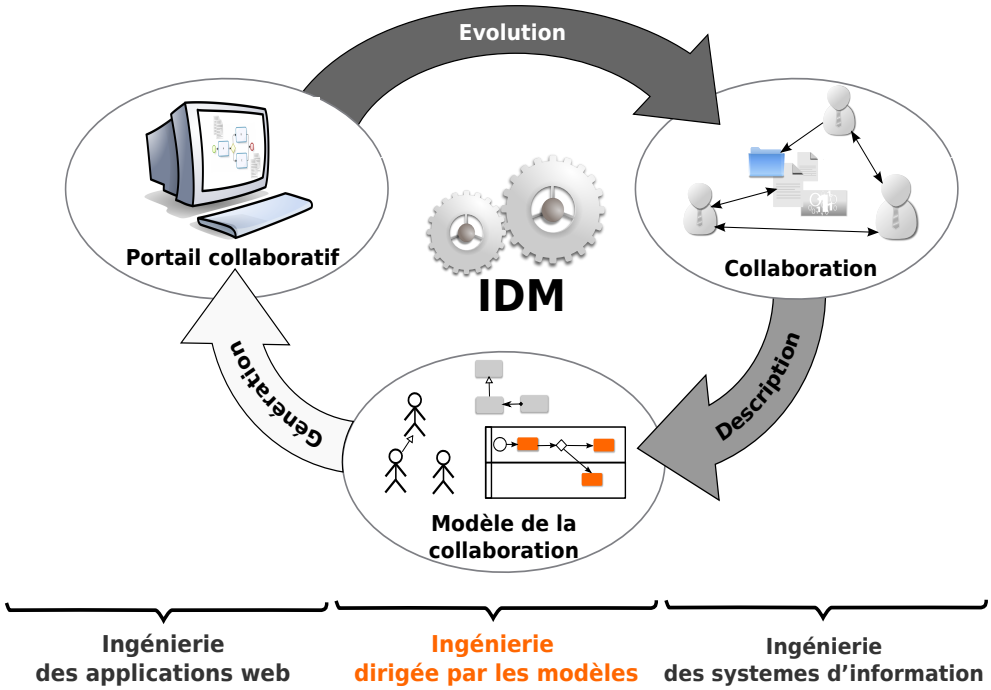


FIGURE 2: L'IDM, une solution pour synchroniser le portail collaboratif au changement du système d'information

1.1 CONTEXTE DE LA THÈSE

La société Ecréall est spécialisée dans le développement, l'intégration et la maintenance des portails collaboratifs sur mesure (c'est-à-dire respectant le métier de ses clients). Experte du système de gestion de contenu (CMS) collaboratif Plone, Ecréall intervient directement en son cœur pour l'adapter aux besoins métier et fonctionnels de ses clients, ceci en adoptant une approche IDM dans le développement de ces applications Web. Pour cela, Ecréall se base sur l'application ArchGenXML [110] lui permettant de capturer les besoins et de concevoir les applications à l'aide de modèles ad-hoc. À partir de ces modèles, Ecréall génère un code correspondant aux besoins métier qui sera ensuite intégré à l'environnement Plone. Ce code est sous forme de squelette qui doit être complété par le développeur. Cette approche IDM permet à Ecréall de fournir des applications robustes et fiables dans des temps très courts. Elle permet aussi de réutiliser partiellement les modèles développés. Forte de son expérience, Ecréall a fait le choix de continuer à innover dans cette approche IDM en formalisant ces différents métamodèles, notamment ceux de

haut niveau, en permettant de capturer de nouveaux modèles, par exemple les processus métier des clients, afin de rendre le modèle accessible aux acteurs métier. Ecréall cherche à automatiser la conception et la génération de portails spécifiques aux métiers de ses clients - cela en modélisant à un haut niveau d'abstraction les besoins de collaborations - puis en générant automatiquement le portail collaboratif opérationnel. Aucune intervention n'est nécessaire sur le code généré, d'ailleurs les modifications de ce code sont proscrites : elles doivent passer par le modèle de haut niveau.

À la recherche d'une réponse à sa volonté de rendre accessible la modélisation des portails collaboratifs par les acteurs métier et de générer entièrement l'application web correspondante, la société Ecréall, en collaboration avec l'équipe DaRT du laboratoire LIFL, a mis en place un travail de recherche portant sur la modélisation à haut niveau d'abstraction et la génération complète de code pour les portails collaboratifs. Ecréall profite ainsi des compétences et de l'expérience de l'équipe DaRT dans le domaine de l'IDM. La collaboration s'est concrétisée par l'établissement de ma thèse en convention CIFRE. Dans le cadre de cette convention, l'équipe DaRT s'est chargée de mon encadrement sur les aspects théoriques de l'étude. La société Ecréall, quant à elle, a participé très activement aux développements, à l'expression des besoins et à l'évaluation des solutions proposées.

Le travail présenté dans ce manuscrit est le fruit de cette collaboration université-entreprise qui a permis à Ecréall de se voir proposer des réponses théoriques à sa stratégie ainsi qu'un prototype lui permettant déjà de générer des portails collaboratifs opérationnels. Cette première brique a permis d'identifier plusieurs problématiques adjacentes et permit, ainsi, aux deux équipes de mettre en place des projets de nouvelles collaborations.

1.2 TRAVAIL PRÉSENTÉ DANS CETTE THÈSE

Le but de ce travail est de mettre en place un environnement de modélisation et de génération de code complet pour les portails collaboratifs. Cette solution repose sur la définition du métamodèle MACoP (Modeling and Analysis of Collaborative Portal) et se distingue par le fait qu'elle s'appuie sur une modélisation des entreprises centrée sur les processus métiers comme modèle de haut niveau afin de générer un portail collaboratif. Cette modélisation se concentre uniquement sur le métier, et ne contient aucun artefact technique en relation avec la technologie cible visée. Cette modélisation est divisée en « points de vue » concentrés chacun sur un problème particulier. Dans cette modélisation nous essayons de répondre aux critères que nous avons identifiés en nous appuyant sur des techniques comme l'imbrication des processus métier. Le métamodèle MACoP n'est donc pas destiné uniquement aux informaticiens, mais

aussi, et surtout, aux non-informaticiens, plus précisément aux différents acteurs de l'entreprise. MACoP est accompagné par un processus de production de logiciels (c'est-à-dire un processus décrivant l'enchainement des transformations de modèles) permettant de passer d'un modèle décrit par MACoP au code du portail collaboratif opérationnel. Un moteur d'exécution de processus de production (PMS+) interactif a été développé dans le cadre de cette thèse afin d'automatiser l'enchainement des transformations du processus de production de logiciels.

1.3 ORGANISATION DE LA THÈSE

- **Deuxième partie (Positionnement) :**
  L'état de l'art ainsi que l'analyse associés à cette thèse sont présentés dans cette partie. Sachant que nos travaux portent sur la génération de code de portails collaboratif à partir d'un modèle exprimant des exigences métier, considérées comme un haut niveau d'abstraction, nous nous sommes orientés vers trois domaines d'ingénierie afin de mieux cerner les problématiques traitées dans cette thèse. Nous abordons, de ce fait, en trois chapitres le domaine de l'ingénierie des systèmes d'information, celui de l'ingénierie dirigée par les modèles et enfin le domaine de l'ingénierie des applications web comme présenté sur la figure 2.
  - **Chapitre 2 (Modélisation des Systèmes d'Information des entreprises) :**
    Dans ce chapitre nous décrivons d'une manière générale l'évolution de la modélisation des entreprises et sa relation avec celle des systèmes d'informations. Nous avons, ainsi, identifié quelques cadres de référence pour la modélisation des entreprises qui ont porté notre attention et nous ont inspiré pour la modélisation des portails collaboratifs. Nous avons, aussi, introduit dans ce chapitre les différents concepts permettant de comprendre les enjeux de cette thèse, comme la notion de système d'information, la notion de processus métier ou la notion de collaboration. L'étude du domaine de l'ingénierie des systèmes d'information nous a permis de profiter de la culture développée dans ce domaine et d'adopter l'ontologie des acteurs métier afin de rendre la modélisation des portails collaboratifs plus accessible pour eux.
  - **Chapitre 3 (L'ingénierie dirigée par les modèles (IDM)) :**
    Dans ce chapitre, nous présentons l'IDM et ses différents concepts de base. Nous discutons, aussi, la notion de processus de production de logiciels permettant l'enchainement de nos transformations de modèles. Une classification des différents types de processus de production est proposée afin de positionner nos travaux. Enfin, nous montrons les différentes problé-

matiques que nous avons identifiées par rapport aux différents outils d'enchainement des transformations de modèles dans une approche IDM.

- **Chapitre 4 (Ingénierie des applications web) :**
  Dans ce chapitre nous présentons le travail fait dans le domaine des modélisations des applications web. Un bref historique de l'ingénierie des applications web y est présenté. Par rapport à la notion de collaboration, nous avons proposé une classification des applications web, ce qui nous a permis de positionner les portails collaboratifs. Dans ce chapitre nous avons, aussi, établi un état de l'art des différentes approches existantes pour la modélisation des applications web. Enfin, nous montrons, dans ce chapitre, que ces approches sont peu adaptées pour la modélisation des applications web type portail collaboratif.
- **Chapitre 5 (Les limites des approches existantes) :**
  Dans ce chapitre nous discutons des limites des approches existantes en nous basant sur la notion d'ergonomie. Ainsi nous avons appliqué la notion d'ergonomie sur les métamodèles ce qui nous a permis d'identifier les différents critères permettant d'avoir un métamodèle ergonomiquement efficace par rapport aux acteurs métier. Les critères identifiés sont ensuite détaillés et les limites des approches existantes sont montrées selon ces critères.

- **Troisième partie (Conception des portails collaboratifs) :**
  Cette partie décrit la contribution de cette thèse. Elle est composée de deux chapitres. L'un, principal, décrivant le métamodèle et les techniques de modélisation adoptés pour les portails collaboratifs. Le second parle davantage des techniques de génération de code et des processus de production de logiciels.
  - **Chapitre 6 (Modélisation des portails collaboratifs) :**
    Dans ce chapitre nous décrivons le métamodèle MACoP. Ainsi, nous détaillons les différents concepts ajoutés afin de modéliser à un haut niveau d'abstraction les portails collaboratifs. Les techniques de modélisation présentées ainsi que les différentes extensions ajoutées nous permettent de créer une rupture avec les approches de modélisation des applications web existantes. MACoP répond aux différents critères que nous avons identifiés dans le chapitre 5.
  - **Chapitre 7 (Du modèle vers l'application) :**
    Dans ce chapitre nous présentons notre outil de modélisation et d'exécution des processus de production de logiciels PMS+. Notre processus de production de logiciels est aussi présenté dans ce chapitre. Ainsi, une brève description des différentes transformations nous permettant de générer nos portails collaboratifs est donnée.

- **Quatrième partie (Validation expérimentale) :**
  - **Chapitre 8 (Evaluation) :**
    Dans ce chapitre Nous présentons une application de notre approche sur un exemple illustrant les différentes techniques évoquées dans les parties précédentes. L'exemple que nous proposons illustre une vision simplifiée du métier d'Ecréall ainsi que l'usage de l'outil « Tracker » dans le cadre d'une démarche agile de gestion de projet informatique. Nous finissons ce chapitre par une synthèse montrant les différences par rapport aux solutions existantes selon les critères identifiés dans le chapitre 5.
- **Cinquième partie (Conclusion et perspectives) :**
  - **Chapitre 9 (Conclusion et perspectives) :**
    Dans ce chapitre nous présentons une synthèse des travaux effectués ainsi que les différentes ouvertures identifiées qui présentent le sujet de nos travaux futurs et ceux en cours.

Deuxième partie

POSITIONNEMENT

2

MODÉLISATION DES SYSTÈMES D'INFORMATION DES ENTREPRISES

Sommaire

La modélisation est une technique qui permet de structurer des pensées et des connaissances qui appartiennent à un domaine de réflexion identifié afin d'en avoir une vision simplifiée et ainsi en gérer la complexité. La structuration des idées joue un rôle stimulateur et permet de produire d'autres idées ce qui contribue à enrichir le modèle initial. L'entreprise a pris conscience de cet avantage et a tiré profit de la modélisation des entreprises afin d'implanter ses différents systèmes. D'une manière générale, un système au sens large est un ensemble d'éléments qui interagissent entre eux selon un enchaînement d'actions organisées et en vue d'atteindre un but bien défini. L'entreprise est composée de plusieurs systèmes assurant son fonctionnement. La modélisation des entreprises permet d'avoir des modèles décrivant des vues, souvent, métiers des systèmes de l'entreprise et se situe à un niveau macroscopique (c'est-à-dire une description à gros-grain).

Avec l'évolution de l'informatique, les systèmes de l'entreprise ont pris une position stratégique et sont devenus inévitables. Nous pensons que, actuellement, la volonté permanente de l'entreprise de dématérialiser et automatiser son métier a réduit le rôle de l'homme aux décisions créatives et stratégiques [127] impliquant le sens et les émotions que la machine ne peut éprouver actuellement. Cette automatisation et dématérialisation du métier de l'entreprise a augmenté l'efficacité de l'homme et lui a permis de se concentrer sur sa valeur ajoutée. Pour satisfaire ce besoin de dématérialisation et au-

La dématérialisation consiste à transformer les documents physiques de l'entreprise en format numérique afin d'être générés automatiquement.

tomatisation, les informaticiens transforment les besoins métier en une application informatique. Dans les faits, ces besoins métier ne sont pas toujours respectés [25]. Le plus souvent, cela est dû à un manque de communication entre les acteurs métier et les informaticiens. Les acteurs métier sont les personnes impliquées dans l'expression des besoins métier et les informaticiens sont les personnes impliquées dans la traduction des besoins métier en une application opérationnelle. Pour résoudre ce manque de communication, les acteurs du génie logiciel proposent la modélisation des systèmes d'information de l'entreprise en suivant une vision informatique de l'entreprise. Ainsi, on trouve une vision métier de l'entreprise concrétisée par la modélisation des entreprises et une vision informatique concrétisée par la modélisation des systèmes d'information dans le but d'en produire des logiciels. Aujourd'hui, les informaticiens tirent profit de la modélisation des entreprises en remontant en abstraction et en adoptant le langage adopté par la modélisation des entreprises [91].

Avec la complexité des domaines d'activité et l'augmentation de la compétitivité, les entreprises deviennent de plus en plus spécialisées. De ce fait, elles perdent en couverture et deviennent souvent incapables de fournir un produit finalisé sans collaborer avec d'autres entreprises. La collaboration est devenue une solution avantageuse afin de monter en puissance et de continuer à exceller dans sa spécialité [67]. Ainsi, les relations d'une entreprise sont passées de relations relativement figées (c'est-à-dire que les relations collaboratives avec les autres entreprises ne changent pas) à des relations en constante évolution (c'est-à-dire que les relations collaboratives avec les autres entreprises changent souvent). Dans ce cas, le système d'information collaboratif et le système informatique collaboratif doivent refléter cette évolution, si possible au même rythme que les changements de collaboration au sein de l'entreprise. Une application informatique doit donc être évolutive et adaptée aux besoins métier de la collaboration.

Le but de ce chapitre est de mettre en contexte nos travaux par rapport aux besoins de collaboration de l'entreprise. Dans la section 2.1, nous donnons un aperçu de l'évolution de la modélisation des entreprises jusqu'à sa normalisation. Cela nous permettra d'adopter le langage des entreprises afin de modéliser nos portails collaboratifs. Pour cela, nous nous sommes intéressés à certaines approches et normes, comme CIM-OSA [74] et UEML [7]. Dans la section 2.2 nous donnons la définition du Système d'Information collaboratif en définissant d'abord le système d'information puis la notion de collaboration. Dans la même section, nous parlons brièvement des différentes approches de modélisation des systèmes d'information et de son évolution jusqu'aux approches actuelles centrées sur les processus métier. Sachant que notre modélisation des portails collaboratifs est centrée sur ces processus, dans la section 2.3 nous définissons en détail le concept de processus métier. Puis dans la section 2.4

nous parlons de la position de ces processus dans la modélisation des systèmes d'information. Enfin dans la section 2.5 nous parlons des portails collaboratifs et leurs utilités ainsi que les problématiques associées à leur développement.

2.1 MODÉLISATION DES ENTREPRISES

La bonne compréhension du fonctionnement interne de l'entreprise mène à une communication efficace entre les différents acteurs qui la constituent. C'est pourquoi les entreprises ont structuré leur organisation autour de processus métier décrivant leurs fonctionnements sous forme d'enchainement d'actions, définis aussi clairement que possible, formalisés et autour desquels les différents métiers communiquent. Cela leur permet d'avoir une vision globale et précise de leur métier et ainsi d'améliorer et optimiser leurs processus métier.

Plusieurs courants de travaux scientifiques accompagnent cette forme de structuration, et des langages de modélisation, souvent graphiques, ont été proposés pour obtenir une représentation formelle du fonctionnement de l'entreprise. La modélisation des entreprises est certainement l'une des formes de structuration les plus matures. Cette modélisation a suivi un schéma classique allant de la diversification à la standardisation en passant par l'unification. En effet, dans les années 70 le projet ICAM (Integrated Computer Aided Manufacturing) par la US Air Force propose la suite IDEF (Icam DEFinition) [56] reconnaissant différentes vues pour la modélisation de l'entreprise allant de la vue informationnelle (IDEF1) à la vue fonctionnelle (IDEF0/SADT) en passant par la vue dynamique (IDEF12).

Les débuts des années 80 ont vu la naissance de la méthode GRAI [32] développée à l'Université de Bordeaux I. L'originalité de cette méthode est qu'elle s'appuie sur un modèle du système décisionnel de l'entreprise (la grille GRAI). Grâce à la richesse de ses langages de modélisation, elle permet, en s'appuyant sur une démarche structurée, l'évolution par étape d'un système tout en assurant, à travers des indicateurs, la possibilité d'évaluer ses différentes vues. Dans la méthode GRAI la spécification des besoins passe par deux phases, une phase orientée utilisateur où les besoins sont transformés sous forme de spécifications fonctionnelles et informationnelles et une phase orientée technologie où les spécifications des besoins sont transformées en spécifications des technologies, permettant ainsi la mise en œuvre du système. Issue des travaux effectués dans le laboratoire GRAI, la méthode GIM (GRAI Integrated Methodology) a vu le jour en apportant une diversification de présentation (GRAI, MERISE,IDEF0) afin de représenter les différentes vues d'un système. La méthode GIM complète la méthode GRAI par l'utilisation de la méthode MERISE et IDEF0.

La méthode CIM-OSA (Computed Integrated Manufacturing, Open System Architecture)[74] a été développée au début des années 90. Son but principal était de permettre la modélisation centrée sur les processus métier pour les entreprises et de proposer un environnement permettant de supporter les différentes opérations des systèmes basés sur cette modélisation. CIM-OSA présente une architecture de référence appelée cube de CIM-OSA. Trois axes sont identifiés : le premier axe (dérivation) présente les différentes phases dans un projet d'entreprise allant de l'identification des besoins jusqu'à l'implémentation ; le deuxième axe (génération) propose de modéliser l'entreprise suivant des points de vue différents (informationnel, fonctionnel, ressource et organisationnel) comme illustrés à la figure 3 ; le troisième axe (instanciation) décrit la méthodologie à suivre pour avoir un modèle spécifique à l'entreprise à partir de modèles partiels exprimés en termes de modèles génériques de base.

L'approche de modélisation des processus métier adoptée par CIM-OSA peut être appliquée d'une manière uniforme sur tous les processus de l'entreprise depuis les processus industriels, jusqu'aux processus de gestion et de contrôle. Une approche modulaire et hiérarchisée est proposée par CIM-OSA afin de classer les activités de l'entreprise par « Domaine » décrit à travers des « Processus Domaines » eux-mêmes détaillés en « Processus Métier ». Ceux-ci sont finalement décrits en « Activités » détaillées en « Opérations fonctionnelles ». La figure 3 illustre cette hiérarchisation dans la vue fonctionnelle du métamodèle CIM-OSA.

La méthode ARIS (The Architecture of Integrated Information Systems) [125] est mise sur le marché en 1992. Cette méthode propose les différentes phases à suivre pour décrire les besoins d'un système d'information. Pour ce faire, différentes vues sont proposées (organisationnelle, de données, fonctionnelle et contrôle).

La méthode ARIS donne une très grande importance aux processus métier de l'entreprise et permet de les décrire à différents niveaux d'abstraction.

La multiplication des méthodes et des normes pour la modélisation des entreprises a conduit à de nombreux travaux tentant d'uniformiser et de standardiser les différents aspects de la modélisation des entreprises (modèles, outils et technologies). Parmi ces travaux, citons notamment ATHENA (Advanced Technology for interoperability of Heterogeneous Enterprise networks and their Applications) [11] qui a été développé dans le but de proposer un cadre de réponse aux besoins d'interopérabilité entre entreprises. Sur le même thème, GERAM [15] a vu le jour pour une généralisation, fondée sur les méthodes les plus abouties comme CIM-OSA et GIM. GERAM a ensuite conduit à une normalisation des méthodes d'ingénierie d'entreprise. Cette normalisation a commencée au début des années 90 pour prendre fin en 2002 et aboutir, *in fine*, à la norme ISO ENV 40003 [57].

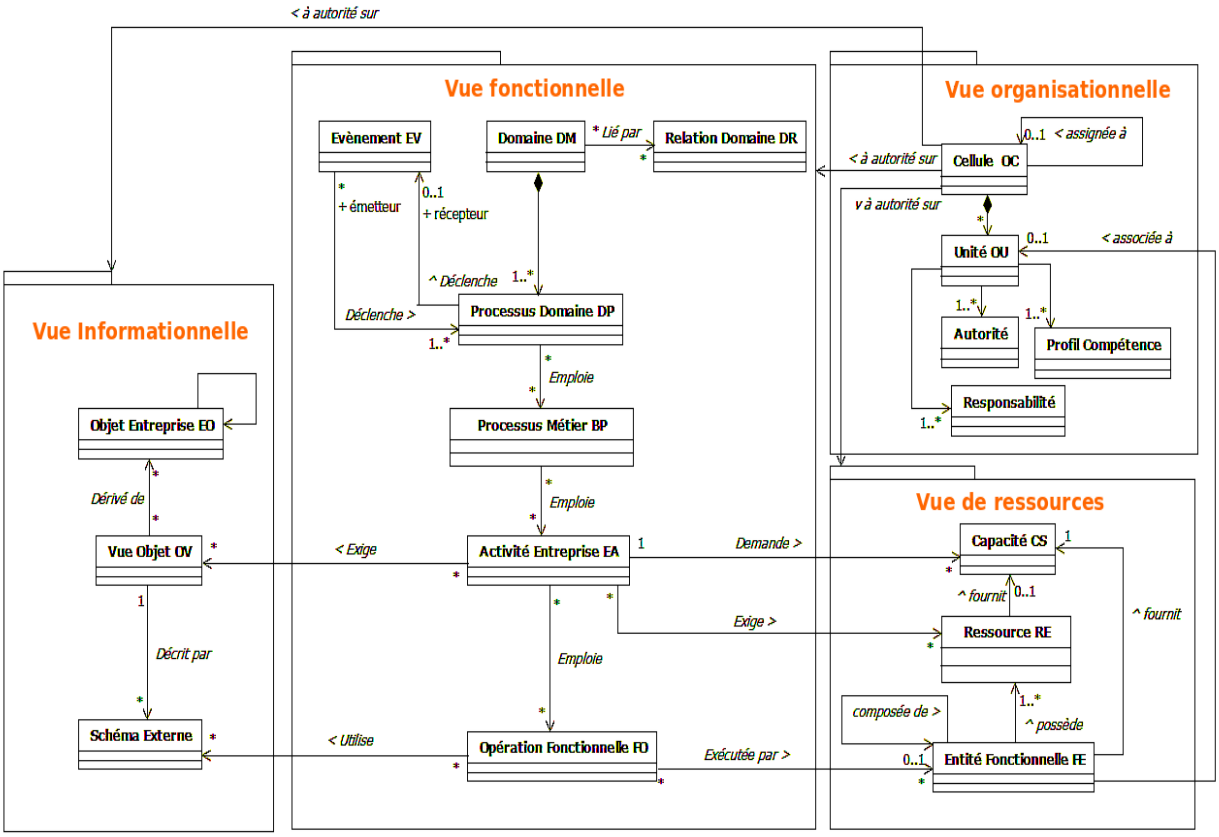


FIGURE 3: Le métamodèle CIM-OSA

Cette norme décompose la modélisation des entreprises en différents « point de vue de modélisation ». Un point de vue de modélisation est une vision particulière de l'entreprise qui met en lumière certains aspects, en augmentant leur niveau de détails, et rend transparents les autres, en diminuant leur niveau de détails. C'est une perspective particulière pour décrire, puis analyser, une même entreprise au moyen du modèle [27]. Ceci permet une communication plus efficace entre les différents acteurs de l'entreprise.

La norme ENV 40003 reconnaît quatre points de vue de modélisation :

- **la vue fonctionnelle** fournit une représentation des processus métier de l'entreprise.
- **la vue informationnelle** fournit une représentation structurée d'un ensemble d'informations de l'entreprise et des relations de dépendance entre ces informations ;
- **la vue des ressources** fournit une représentation de l'ensemble des moyens nécessaires pour mettre en œuvre les activités de l'entreprise ;
- **la vue organisationnelle** fournit une représentation de l'organisation structurelle de l'entreprise.

D'autres normes pour l'entreprise ont été proposées traitant chacune une problématique bien définie. Ce qui a suscité une tentative de normalisation d'échanges entre les différentes modélisations. Ainsi un programme de collaboration a eu pour objectif de définir une interface standardisée entre les outils de modélisation d'entreprise basés sur des modèles différents. Ces travaux ont finalement conduit au métamodèle UEML [7] qui permet certains degrés de compatibilité entre les outils. En résumé, UEML est un standard de modélisation des entreprises assurant l'interopérabilité entre la multitude de métamodèles dédiés à la modélisation des entreprises. La figure 4 illustre une partie du métamodèle UEML selon les quatre points de vue de la norme ISO ENV 40003. Nous trouvons, ainsi, la vue informationnelle qui décrit les objets de l'entreprise parmi lesquels on trouve les ressources, par exemple humain ou machine ; la vue fonctionnelle avec les processus métier déclenchables par des événements et qui comprennent les activités ; la vue de ressources qui décrit les rôles avec leurs compétences et leurs qualifications et enfin la vue organisationnelle qui décrit les unités organisationnelles qui composent l'entreprise.

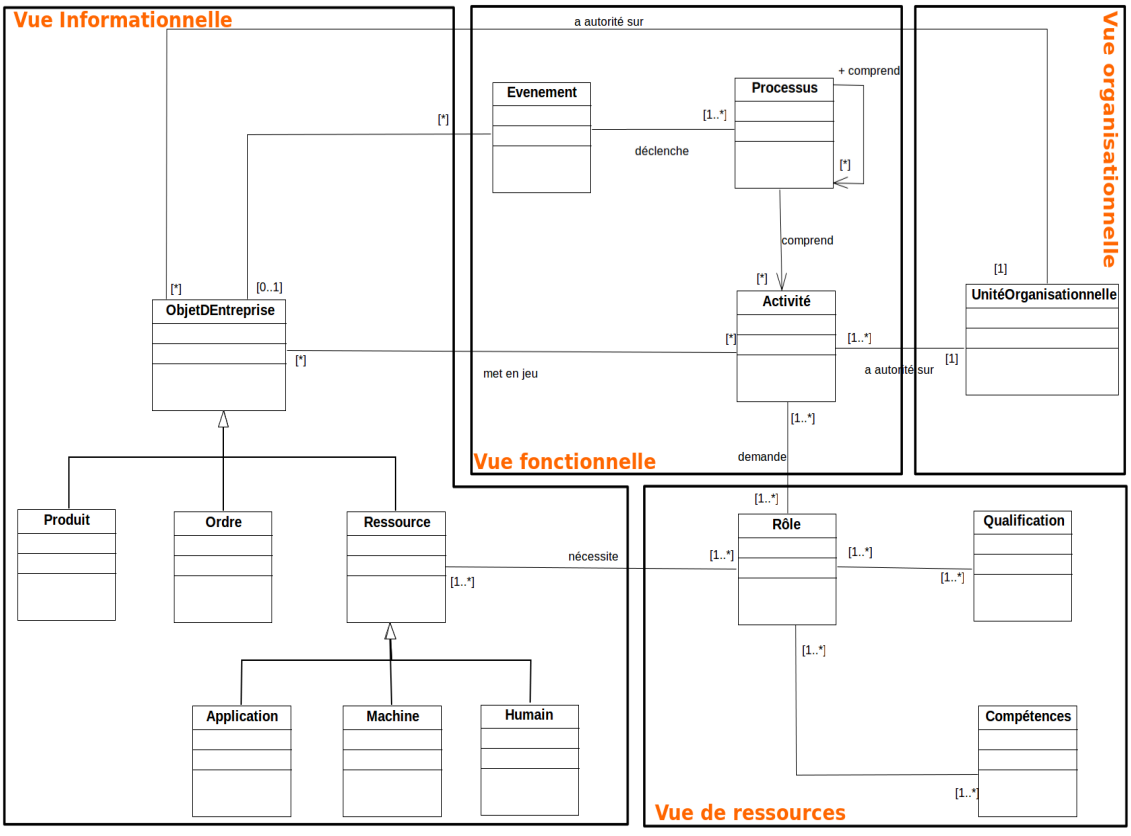


FIGURE 4: Le métamodèle UEML

Dans nos travaux, nous nous intéressons à la modélisation des entreprises car cette dernière est accessible par les acteurs métier. Nous

nous sommes, donc, intéressés aux travaux faits dans CIM-OSA pour sa hiérarchisation du fonctionnement de l'entreprise, comme nous l'avons vu précédemment, ainsi qu'au métamodèle UEML pour son approche unificatrice et les concepts d'objet d'entreprise de la vue informationnelle et les rôles de la vue des ressources. Nous avons aussi retenu les travaux de synthèse relatifs à la norme ISO ENV 40003 et plus précisément la notion de « point de vue de modélisation » que nous retrouvons dans les métamodèles UEML et celui de CIM-OSA.

La modélisation des entreprises avec ces différentes vues cherche à structurer le fonctionnement de l'entreprise afin de le clarifier, l'expliquer et l'analyser. Ainsi, le modèle constitue un support invitant les différents acteurs métier à la réflexion dans le but de faire évoluer le système d'une manière générale. Ce modèle ne présente donc qu'une vue métier de ce qui forme l'entreprise, et se situe à un niveau d'abstraction élevé. Avec l'avènement des nouvelles technologies et spécialement l'informatique, l'entreprise est devenue de plus en plus réactive et en constante évolution afin de rester dans la course avec la concurrence. L'entreprise a de plus en plus besoin de croître et de se développer. Pour répondre à ce besoin, elle s'appuie toujours davantage sur des outils/programmes informatiques.

En effet, ces derniers permettent d'assister et de simplifier le travail des acteurs de l'entreprise à travers l'automatisation de leurs métier ainsi qu' à travers la gestion et le traitement des informations qui constituent la valeur ajoutée de l'entreprise. Le besoin de développer des applications respectant mieux le métier de l'entreprise, notamment sur toute la durée d'un processus métier, est devenu nécessaire.

Dans un souci d'efficacité et de respect du métier de l'entreprise, les informaticiens ont opté pour une modélisation plus informatique afin de produire ces applications. Cela pour faciliter le passage d'une vue métier, chargée d'intentions non concrétisées, à une vue plus formelle pour produire des applications sur mesure. Cette modélisation produit des modèles de systèmes d'information orientés informatique décrivant une solution technique du logiciel, solution en général difficilement compréhensible par les acteurs métier. Les métamodèles permettant de décrire ces modèles sont caractérisés par une expressivité élevée. Cela afin de décrire en détails le système à implémenter. Pour rendre les modèles du système d'information plus accessible par les acteurs métier, les informaticiens ont cherché à remonter plus en abstraction afin de s'approcher de la modélisation des entreprises [91]. Nous pensons qu'une solution intermédiaire entre la modélisation des entreprises et celle des systèmes d'information est donc nécessaire afin de garder un haut niveau d'abstraction et une expressivité satisfaisante. Cette solution doit permettre la production des logiciels sur mesure pour l'entreprise.

2.2 SYSTÈME D'INFORMATION COLLABORATIF : DÉFINITION ET MODÉLISATION

Dans la section précédente, nous avons présenté la modélisation des entreprises ainsi que les différentes approches auxquelles nous nous sommes intéressés. Dans cette section nous abordons la notion de système d'information collaboratif afin d'introduire les concepts utilisés dans cette thèse et de positionner les portails collaboratifs dans ce contexte. Afin de définir le système d'information collaboratif, nous commençons par définir la notion de système d'information et proposer notre propre vision de celui-ci. Ensuite, nous détaillons la notion de collaboration. Cette dernière peut avoir plusieurs niveaux de maturité. Un positionnement des portails collaboratif par rapport à ces niveaux est donné. Enfin, nous parlons des approches de modélisation des systèmes d'informations. Nous verrons, alors, que cette modélisation devient de plus en plus centrée sur les processus métier.

2.2.1 *Qu'est-ce qu'un système d'information ?*

Les connaissances définissant les systèmes ont atteint un niveau de complexité, structurelle et fonctionnelle, tel qu'il devient difficile de les maîtriser sans les décomposer. Tout en suivant le schéma classique de maturation des connaissances, les systèmes d'une manière générale (comme ville, machine ou entreprise) ont été définis dans le but de les comprendre et d'améliorer leurs fonctionnements.

Dans [118], Rosnay and Al. proposent l'approche systémique qui se définit comme une méthodologie permettant de cerner les connaissances par rapport à un système donné afin de mieux agir sur ce dernier. En appliquant cette approche à l'entreprise, on peut modéliser celle-ci en un système d'entreprise composé de trois sous-systèmes (voir figure 5) :

- Le système de pilotage ou de décision qui permet de piloter l'entreprise selon des stratégies et des objectifs bien définis. C'est à ce niveau que les décisions stratégiques de l'entreprise sont prises (Flux de décision).
- Le système opérant qui permet de transformer des entrées (données à l'état primaire) en sorties (produits ou services transformés pour une finalité identifiée). Cela correspond au flux physique.
- Le système d'information qui relie les deux systèmes cités précédemment en assurant l'acquisition et le transfert d'informations afin d'assurer une certaine harmonie entre la prise de décision et la production. Ce système permet de consommer et de produire de l'information (Flux d'information).

Dans l'approche fonctionnelle [94], le système d'information est présenté comme un ensemble de fonctions permettant d'assurer la dispo-

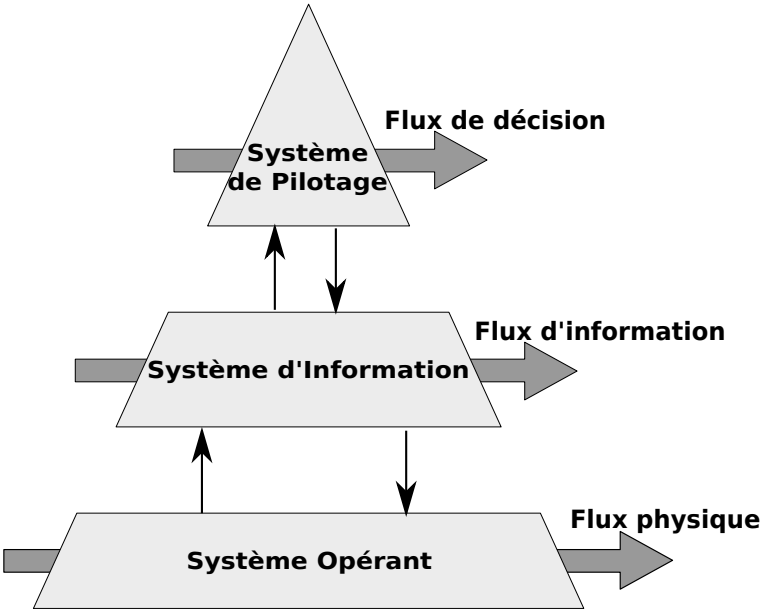


FIGURE 5: Le système d'information selon l'approche systémique

nibilité de la bonne information pour la prise des décisions. Selon cette approche, le système d'information n'a pas de limite et s'étend sur les deux autres systèmes décrits dans l'approche systémique. En effet, les décisions, les hommes, les produits et les activités sont considérés comme faisant partie du système d'information.

Dans l'approche structurelle [94] le système d'information est composé de deux sous- systèmes : le système de traitement d'information et le système informatique. Comme illustré sur la figure 6, le système de traitement de l'information décrit comment les acteurs collaborent en agissant sur les informations pour atteindre un objectif de production selon des processus métier bien définis. Cette approche met l'accent sur la notion de processus et sa relation avec les acteurs et les informations. Le système informatique quant à lui assure l'infrastructure matérielle et applicative afin de permettre le fonctionnement du système de traitement de l'information.

La figure 7 montre une autre alternative de l'approche structurelle [94]. Cette alternative met plus en avant le système d'information en le confondant avec le système de traitement de l'information. Dans cette approche, le système informatique est considéré comme indépendant.

L'approche structurelle ne spécifie pas la limite entre ce qui est la partie physique de l'entreprise, par exemple un ordinateur ou une personne, et la partie théorique (c'est-à-dire l'information sur la partie physique) de l'entreprise, par exemple la configuration d'un ordinateur ou l'identité d'une personne. Dans cette approche la partie physique et la partie théorique sont réparties sur le système informatique et le système d'information. Par contre dans l'approche fonctionnelle, la partie physique décrit par le système opérant est séparée

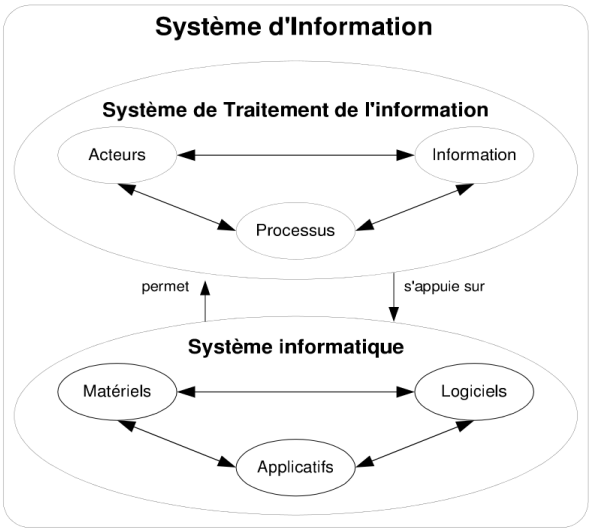


FIGURE 6: Système d'information selon l'approche structurelle [94]

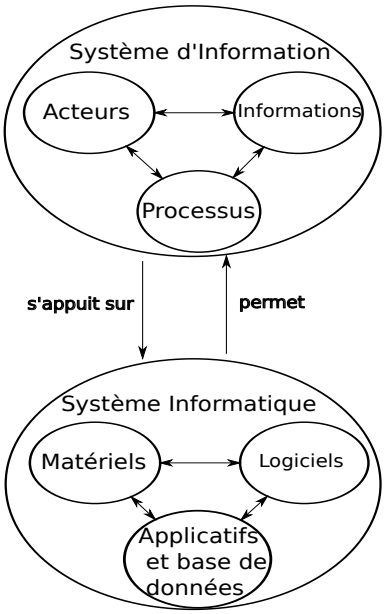


FIGURE 7: Système d'information et système informatique selon l'approche structurelle [94]

de la partie théorique décrite par le système d'information et le système du pilotage.

Pour nos travaux, nous proposons, pour le SI, une définition hybride entre l'approche fonctionnelle et l'approche structurelle. Cette définition décompose l'entreprise en deux systèmes qui sont le système d'information et le systéme physique. Dans notre définition, le système physique ne peut exister sans le système d'information et le système d'information ne peut exister sans le système physique : dans notre définition, une information est un ensemble de connaissances. Cette information porte sur l'existant matériel, par exemple la

configuration d'un ordinateur ; ainsi que non matériel, par exemple un théorème mathématique d'un système. L'information peut porter sur la description des processus de contrôle ou de gestion. Elle peut, aussi, décrire les prises de décisions. Enfin, elle peut être à un niveau microscopique (c'est-à-dire une information détaillée à fin-grain) ou macroscopique (c'est-à-dire une information générale à gros-grain).

Dans notre définition, le système d'information est constitué de l'ensemble d'informations sur l'existant matériel et non matériel d'une entreprise. Ce système donne à l'entreprise sa valeur ajoutée. Le système physique, quant à lui, permet le fonctionnement du système d'information afin de consommer, de traiter et de produire de l'information (incluant la prise de décisions et la production). Ce système inclut les humains ainsi que les machines.

Lorsqu'il est complexe, le système d'information est décomposable en plusieurs sous systèmes, qui collaborent entre eux.

2.2.2 *Qu'est-ce qu'une collaboration ?*

Le dictionnaire Larrousse définit une collaboration comme suit : *Action de collaborer, de participer à une œuvre avec d'autres*. Cette définition sous-entend en général l'action de personnes et se concrétise par l'exécution de certains processus pour produire de la valeur ajoutée. Par exemple, des personnes peuvent collaborer par l'échange de documents, suivant un cycle de vie bien identifié.

Dans le cadre informatique, une collaboration se fait aussi bien entre humains qu'entre humains et machines ou entre machines. On parle alors d'acteurs de la collaboration, acteurs qui peuvent être aussi bien des humains que des machines.

Ainsi, une collaboration est l'échange entre deux ou plusieurs entités actives afin d'atteindre un objectif bien déterminé. Une entité peut prendre plusieurs formes à différentes échelles. Elle peut être à un niveau macroscopique, par exemple une entreprise, une organisation ou un pays, ou microscopique, par exemple, une personne. Pour une entreprise nous distinguons la collaboration interne, entre les entités qui la composent, ou externe, avec les autres entreprises, fournisseurs ou clients. Dire qu'il existe une collaboration entre les entreprises n'est qu'un point de vue à une échelle élevée. Cela est équivalent à dire qu'il existe une collaboration entre des entités qui appartiennent à des entités englobantes différentes.

Les entreprises, aujourd'hui, sont devenues de plus en plus spécialisées dans leurs activités. Par exemple, dans le cas du développement d'une application web, différentes entreprises entrent en collaboration : une première société s'occupe de l'aspect métier, une autre de l'aspect ergonomie et identité visuelle et enfin une troisième pour l'hébergement de l'application. Ces différentes entités (prestataires et client) collaborent en suivant des processus dans l'objectif d'obtenir

une application opérationnelle. Cette application permettra, lors de son fonctionnement, aux acteurs de l'entreprise de collaborer entre eux. On voit, sur cet exemple, que chaque entreprise étend son métier à travers ses collaborations. De plus, selon la nature de la collaboration, en termes de relation entre les partenaires, l'entreprise gagne en puissance, en optimisation ou en couverture [67].

Avec les progrès technologiques, les supports permettant de concrétiser et faciliter la mise en place d'une collaboration sont devenus de plus en plus nombreux (téléphone, fax, mail et autre moyen de communication). Cela a permis aux relations entre entreprises de passer d'une collaboration externe souvent figée dans le temps à une collaboration externe plus fluide et variable [67]. Une collaboration peut passer par plusieurs étapes présentant des niveaux de maturations différents, parmi lesquels on peut citer les niveaux suivants [79] :

- Communicant : Ce niveau correspond a la phase de communication. Pendant cette phase les partenaires se découvrent et échangent sur leur méthodologie de travail en s'outillant par des moyens technologiques classiques comme l'email. Cette phase est généralement asynchrone.
- Ouvert : Ce niveau correspond a la phase de confiance où les différents collaborateurs mettent à disposition de leurs partenaires une structure leur permettant de communiquer d'une manière plus fluide (synchrone) dans le but de réaliser des objectifs individuels. À ce stade, il n'existe pas de processus collaboratif identifié.
- Fédéré : Ce niveau correspond a la phase du travail collectif selon des processus de collaboration établis entre les différents partenaires pour atteindre des objectifs communs. Ce niveau et le précédent peuvent être assurés par des moyens technologiques plus adaptés comme les applications orientées servies.
- Interopérable : À ce stade la collaboration devient une partie prenante des différents collaborateurs pour former une seule organisation (virtuelle ou réelle).
  Les outils utilisés dans cette phase sont plus adaptés et plus intégrés. Ils nécessitent souvent la mise en place d'un système complet intégré aux différents systèmes impliqués dans la collaboration.

Selon nous, la maturité de la collaboration et les technologies utilisées évoluent parallèlement. Ainsi, plus la collaboration est mature plus le besoin d'avoir des outils adaptés à la collaboration devient nécessaire.

2.2.3 *Système d'information collaboratif*

Touzi et Al [59, 67] définissent la notion de système d'information collaboratif (SIC) comme étant un système d'information qui inclut

les différentes parties publiques des systèmes d'information des entreprises participantes à une collaboration donnée. Dans cette définition, la collaboration se fait de façon externe aux entreprises impliquées dans la collaboration. Le SIC joue alors un rôle d'interopérabilité entre les différents systèmes d'information de ces entreprises. Un système d'information peut être décomposé en plusieurs sous-systèmes d'information. Dans ce cas, les systèmes d'information collaboratifs internes à l'entreprise permettent l'interopérabilité des différents systèmes d'information d'une même entreprise. Sachant que le système informatique permet le fonctionnement du système d'information, un système informatique collaboratif est un système informatique permettant le fonctionnement du SIC.

La figure 8 montre la relation entre le SIC et les différents systèmes d'information impliqués dans une collaboration externe. Cette figure illustre, aussi qu'un SI peut lui-même être constitué d'un SIC avec ses SI, montrant ainsi la relativité du SIC qui peut être définie en interne à l'entreprise. C'est le cas du SI2 de la figure 8.

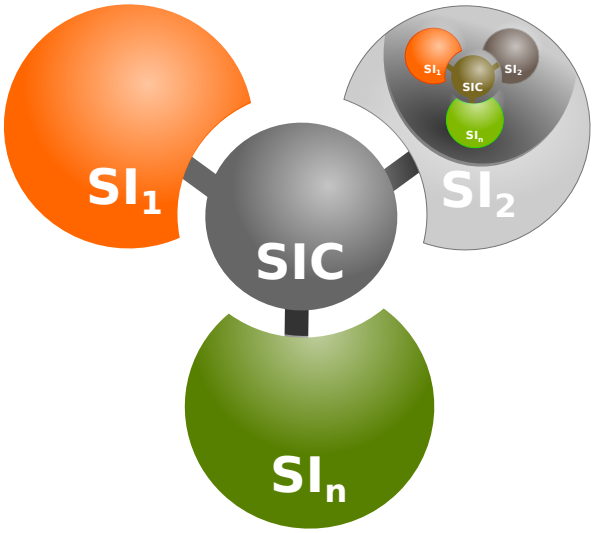


FIGURE 8: Système d'information collaboratif

La limite entre système d'information et système d'information collaboratif est floue. En effet, en nous appuyant sur l'idée que tout est collaboration nous pouvons conclure qu'un système d'information est, par convention, un système d'information collaboratif interne. Un système d'information collaboratif externe, quant à lui, peut évoluer en un système d'information tout court si la collaboration atteint une maturité élevée (c'est-à-dire interopérable).

2.2.4 *Modélisation des systèmes d'information*

Une modélisation purement métier est centrée sur le besoin métier de l'entreprise et non sur les détails techniques et technologiques qui constitue la solution aux besoins métier

Les systèmes d'information ont pris une place importante dans l'entreprise du fait des évolutions technologiques et notamment de l'évolution du domaine informatique. La modélisation du système d'information est passée d'une modélisation purement métier , dans le but d'exprimer et de communiquer via la modélisation des entreprises, à une modélisation avec une vision plus informatique. En effet, avec le besoin de développement des applications informatiques pour les systèmes d'information, une description précise et technique a donnée une vue plus informatique à cette modélisation.

Deux courants méthodologiques majeurs ont vu le jour pour la modélisation des systèmes d'information : les méthodes cartésiennes et les approches systémiques.

Les méthodes cartésiennes [], apparues dans les années 60, se focalisent sur la notion de processus et leur décomposition. Cette approche perçoit le système d'information comme un système de traitement de l'information. Dans cette catégorie, citons par exemple SADT (Structured Analysis Design Technic) [120] une méthode cartésienne apparue dans les années 70 qui permet d'analyser un système sur plusieurs niveaux successifs d'approche descriptive. Cette méthode est fondée sur un formalisme graphique et textuel. Elle permet d'une part de modéliser le problème posé (informatique ou autre) et d'autre part d'assurer une communication efficace entre les différents acteurs impliqués dans la modélisation du système. SADT se focalise sur la description par décomposition hiérarchique des actions. Dans SADT une action consomme des données et produit des données sous le contrôle des données de contrôle. Enfin, ces actions sont assistées par des ressources et outils (mécanisme). Les différents points de vue hiérarchiques permettent une lisibilité importante du modèle et une communication efficace entre les différents acteurs.

Cependant, les méthodes cartésiennes ont plusieurs points faibles [8]. En effet d'après C. FLOYD [46], les méthodes cartésiennes ne sont pas applicables à des systèmes complexes riches en interaction humain-machine.

Les approches systémiques se focalisent sur la description des données à travers le célèbre modèle Entité-Association (EA). Dans cette catégorie nous pouvons citer MERISE [138] dans lequel la dynamique du système d'information est décrite schématiquement par des activités déclenchées par des événements et produisant des événements. Dans cette approche, le flux de contrôle n'est pas décrit et peut être déduit des événements permettant le déclenchement des actions ainsi que les événements produits par les actions. Cette méthode a trouvé un grand succès dans la modélisation et la mise en place des bases de données relationnelles ainsi que la description, dans le but d'analyser les systèmes d'information.

UML (Unified Modeling Language) [104] se veut la continuité et l'unification des approches précédentes. UML introduit notamment la notion de points de vue offrant différentes approches d'un même système. Ceci se traduit par les différents diagrammes d'UML, chacun permettant d'appréhender le système sous un point de vue différent. UML propose une vision plus proche des langages orientés objet que de celui de l'entreprise. UML essaie de réduire la distance entre ces deux mondes afin d'avoir des logiciels de bonne qualité et qui répondent aux besoins métier. Cela reste limité [91] puisque le métier, et particulièrement les processus métier sont décrits de manière souvent éclatée sur tout le modèle après avoir subi une transformation par les concepteurs logiciel afin de décrire les besoins techniques de l'application. Ceci entraîne une difficulté dans la communication entre les acteurs métier et les concepteurs logiciel. Pour résoudre ce problème plusieurs travaux [111] portent sur des extensions à apporter à UML pour permettre la modélisation des processus métier. Parmi ceux-ci, l'OMG propose le métamodèle BPDM (Business Process Definition Metamodel) [101] pour faire évoluer le diagramme d'activité d'UML. Ainsi nous assistons à de nouvelles approches de conceptions de systèmes d'information plus centrées sur le processus métier, pour enfin s'approcher de plus en plus de la modélisation des entreprises. Cela permettra d'améliorer la communication entre les différents participants (acteurs métier et concepteurs logiciel) dans la conception d'un logiciel et une réduction des erreurs de traduction entre ces derniers.

2.3 PROCESSUS MÉTIER : NOTIONS ET CONCEPTS

Le concept de processus métier est central dans notre thèse. Nous allons maintenant détailler les différents concepts et notions y afférant.

Dans une entreprise, un produit est en général le résultat d'un enchaînement d'actions, selon des règles métier, réalisées par des acteurs métier bien identifiés. L'enchaînement de ces actions est appelé un processus métier. Certaines de ces actions peuvent être automatisées dans le but de réduire les temps de production et de simplifier l'interaction des différents acteurs avec le système d'information. Cela permet aux acteurs de se concentrer sur la valeur ajoutée. Selon la Workflow Management Coalition (WfMC) [148] un processus métier est défini comme : « *un ensemble de procédures et d'activités plus ou moins liées qui réalisent collectivement un objectif métier, en général au sein d'une structure organisationnelle définissant des rôles et des relations fonctionnelles. Un processus métier peut être entièrement inclus dans une organisation simple ou peut s'étendre sur plusieurs organisations.* ».

Dans cette définition, les activités peuvent être automatiques exécutées par le système (acteur machine), par exemple la notification

d'un événement, ou manuelles exécutées par les acteurs humains, par exemple la livraison d'un colis.

Un processus métier n'est pas figé dans le temps puisqu'il reflète le métier et les collaborations de l'entreprise qui évoluent constamment. Les processus métier ont un cycle de vie en constante évolution qui est composé de quatre phases comme illustrées à la figure 9, proposée par [51] :

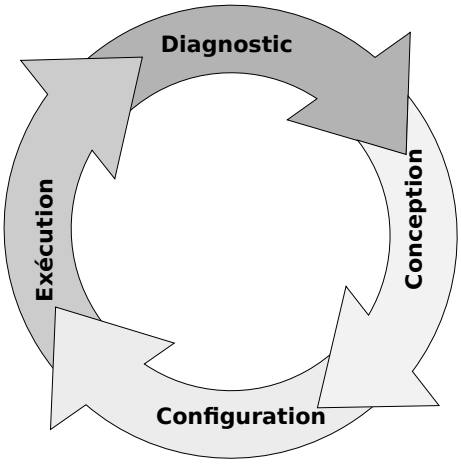


FIGURE 9: Cycle de vie des processus métier [51]

- Conception : Dans cette phase on se focalise sur l'enchaînement logique des activités et leur relation avec les rôles sans se soucier des détails. Cette phase doit rester la plus générique possible.
- Configuration : Dans cette phase on détaille le processus métier en associant les rôles à des acteurs concrets de l'entreprise et en associant une machine ou un programme à une activité. Cette phase est aussi appelée phase d' implantation.
- Exécution : Au cours de cette phase, les activités s'enchaînent les unes après les autres en suivant la définition du processus. Dans ce cas, le processus et ses activités sont instanciés. Ces dernières sont exécutées par les acteurs métier selon leur rôle dans le processus.
- Diagnostic : Cette phase permet d'analyser les processus qualitativement et quantitativement afin de proposer des améliorations possibles pour le processus.

Ce cycle de vie est assisté par un système de gestion de workflow (WMS) permettant aux experts métier de gérer leurs processus métier de bout en bout allant de la définition (Process Definition), qui porte principalement sur la représentation du processus dans une forme qui supporte l'automatisation comme la modélisation, jusqu'à l'exécution interactive avec les acteurs et machines impliquées dans le processus. Dans ce cas, on parle de Workflow qui présente une vision automatisable d'un processus métier.

Selon le WfMC, l'exécution des processus, dans un WMS, est basée sur le concept de *bon de travail*. Pour une activité courante, le système produit un *bon de travail* qui sera dupliqué pour tous les acteurs pouvant l'exécuter, une fois un bon consommé ses copies sont supprimées. Chaque activité consomme et produit des données dans des conteneurs d'entrée et de sortie. Ainsi les données nécessaires à l'exécution des prochaines activités restent disponibles. L'enchaînement des activités est fait en respectant les contraintes d'exécution (post et précondition).

Ce modèle d'exécution est centré sur les activités (ou tâches). Dans ce cas, les activités sont considérées comme des entités explicites directement manipulables par les acteurs. Ces activités sont présentées aux acteurs sous forme d'une liste de tâches à réaliser séparément des données sur lesquelles elles s'appliquent (travailler sur les activités pour agir sur les données). Ce modèle d'exécution est généralement adopté par les solutions BPM. Nous parlons dans ce cas d'orchestration des processus métier.

Selon nous, ce modèle d'exécution ne reflète pas la réalité. En effet, selon le contexte dans lequel l'acteur se trouve et selon son rôle dans l'entreprise, il effectue des actions sur les données de l'entreprise. Dans ce cas, les données sont au centre de l'exécution. Par conséquent, le processus métier est exécuté en arrière-plan. Cette exécution est donc centrée sur les données (travailler sur les données en suivant les processus métier). Cela est plus adapté aux méthodes de travail habituelles.

Par extension, on peut identifier deux familles de processus métier :

- Les processus métier centrés sur les activités (ou tâches) : Cette famille se focalise sur les tâches et permet de décrire d'une manière globale le métier de l'entreprise. Une tâche, dans ce cas, peut prendre plusieurs données comme paramètre d'entrée. Cette approche permet également de décrire l'échange des données suivant les activités de l'entreprise.
- Les processus métier centrés sur les données ou artefacts : Cette famille se focalise d'avantage sur les données et leur cycle de vie, ce qui donne une vue plus explicite sur les étapes par lesquelles ces données passent. Dans [123] [29] les auteurs définissent deux modèles pour cette présentation : le modèle d'informations qui présente les données manipulées pour effectuer une action donnée ; et le modèle de cycle de vie qui définit comment un artefact passe d'un état à un autre (ce modèle peut être implémenté par le diagramme d'état transition d'UML). Assurément, cette vision est informative, mais reste floue pour décrire d'une manière globale le métier de l'entreprise.

Ces deux approches, centrée données et centrée tâches, décrivent deux vues différentes de la vue fonctionnelle d'une collaboration. Selon nous, un processus métier doit permettre de décrire d'une

manière précise le flot d'activités, les données manipulées et les échanges des données entre les différents acteurs du processus. Selon nous, une telle description permet de passer de la vue centrée sur les tâches à la vue centrée sur les données.

La figure 10, extraite de [51], montre la relation entre les différents concepts traités dans cette section. Dans cette figure, un processus métier est défini par une définition de processus. Cette définition est composée de sous-processus métier ou d'activités pouvant être manuelles ou automatiques. Un processus métier est géré par un système de gestion de processus permettant l'orchestration des instances de processus métier incluant une ou plusieurs instances d'activité. Cette dernière est associée à des *bons de travail* autorisant l'exécution de l'action. Une instance de processus métier peut invoquer des applications externes.

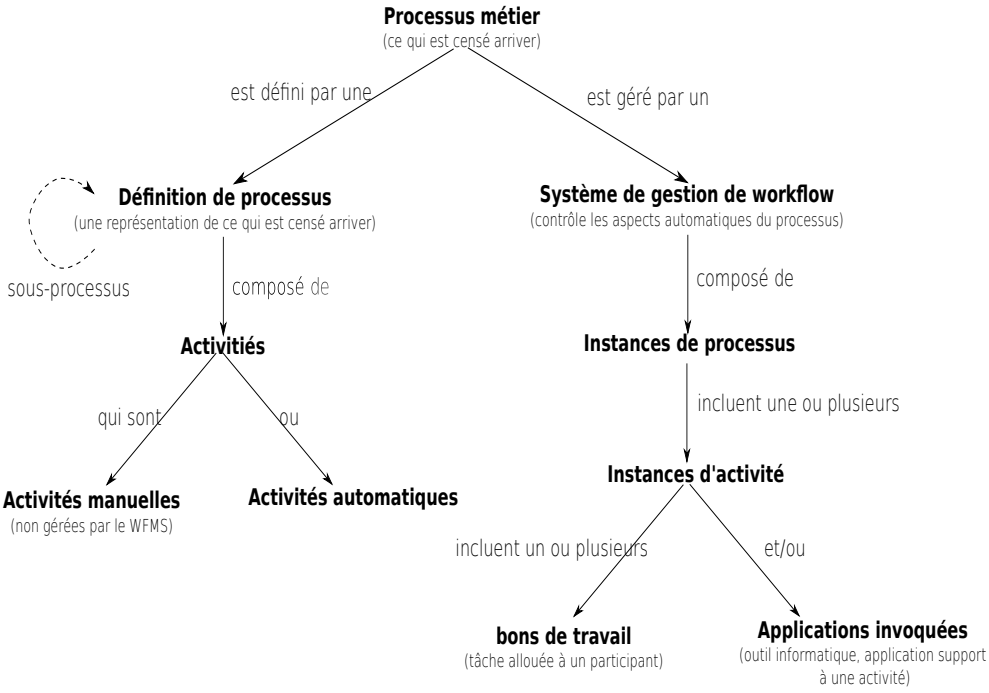


FIGURE 10: Processus métier (les concepts) [51]

Il existe plusieurs métamodèles permettant la modélisation des processus métier. Nous citons à titre d'exemple EPC, les réseaux de Petri, les diagrammes d'activité de UML, BPMN et bien d'autres [51] [73]. Ces métamodèles ont différentes finalités allant de la vérification comme les réseaux de Petri à la conception logicielle pour le diagramme d'activité de UML en passant par les métamodèles purement métier comme BPMN ou EPC.

2.4 POSITION DES PROCESSUS MÉTIER DANS LA MODÉLISATION DES SYSTÈMES D'INFORMATION

La vue fonctionnelle a pris de plus en plus de poids dans la modélisation des entreprises et des systèmes d'information à tel point qu'elle est devenue centrale et joue le rôle de pont entre les différentes vues. Ce pont est concrétisé par les activités qui consomment et produisent des objets de l'entreprise, requièrent un ensemble de ressources pour fonctionner et sont mises sous la responsabilité d'une unité organisationnelle. Cette focalisation sur la vue fonctionnelle a permis d'évoluer d'une description statique du système d'information, permettant de décrire les informations et leur disponibilité, à une description dynamique du système d'information, permettant de décrire comment les informations sont manipulées afin de produire de la valeur ajoutée [95]. Ainsi, nous pouvons distinguer deux vues principales, une vue passive constituée de concepts statiques qui portent de l'information (le Quoi : informations ou objets de l'entreprise, le Qui : Rôle ou unité organisationnelle) et une vue active (le Comment : processus métier). Ces vues permettent de répondre à la question « Comment le Qui agit sur le Quoi ? » mais ne permettent pas de répondre à la question « Comment contrôler le Comment ? »

Puisque le Comment est considéré comme un concept actif qui contrôle et qui n'est pas contrôlé. Le contrôle du Comment permet de décrire des processus de gestion et de supervision de processus identifiés comme le Business Activity Monitoring (BAM). Il permet aussi d'imbriquer les processus afin d'avoir un comportement complexe d'un côté et une modélisation simple, uniforme et accessible d'un autre côté. Dans cette perspective, le Comment est considéré comme le Quoi. De ce fait, nous pensons qu'un processus métier peut se modéliser dans deux vues complémentaires : une vue passive qui appartient à la vue informationnelle et une vue active qui appartient à la vue fonctionnelle.

La position centrale des processus métier dans la modélisation des systèmes d'information a fait apparaître différentes approches pour l'implantation de ces derniers. On trouve, par exemple les approches d'urbanisation des systèmes d'information qui respectent en général trois étapes allant de l'identification des processus métier à l'identification des composants informatiques en passant par l'identification des activités des processus [67]. Ces approches visent à mettre en place un système complet (système d'information et système physique). Une autre approche est celle du Business Process Management (BPM) [72][33] qui vise principalement à définir clairement les processus métier afin de les rendre exécutables par WMS. Ainsi, cette solution vient se greffer sur le système d'information existant et permet d'orchestrer les activités de l'entreprise.

2.5 PORTAIL COLLABORATIF COMME APPLICATION WEB CENTRÉE SUR LES PROCESSUS MÉTIER

Après avoir clarifié la notion de système d'information et de collaboration, nous avons défini la notion de système d'information collaboratif. Ce dernier s'appuie sur le système informatique collaboratif qui se réduit, en partie, à un ensemble d'applications. Un portail collaboratif est l'une de ces applications. Cet outil informatique, généralement constitué d'un serveur web, réalise l'interface entre les différents systèmes d'informations impliqués dans les collaborations. Ce portail permet à un acteur donné d'agir sur les informations d'une collaboration en suivant quatre grands périmètres fonctionnels qui sont la gestion de la connaissance, l'espace collaboratif métier, la plate-forme communautaire et la publication.

Selon nous, un portail collaboratif peut avoir deux formes :

- Il peut être implémenté sur un système (constitué du système d'information et système informatique) déjà existant. Le portail collaboratif est dans ce cas, considéré comme une application d'orchestration de système d'information. Cette solution constitue un support d'interopérabilité limité aux différents systèmes (systèmes d'information et systèmes informatique) impliqués dans la collaboration.

  Par conséquent, cette forme de portail collaboratif peut être envisagée à un niveau de maturité fédéré de la collaboration et reste moins adaptée à celui du niveau interopérable. La majorité des solutions BPM qui existent aujourd'hui sur le marché comme BonitaSoft [18], Intalio [45] ou WebRatio [21] adoptent cette forme. Nous détaillerons ces approches dans le chapitre 4 de ce manuscrit.
- La deuxième forme préconise de migrer les différents systèmes d'information impliqués dans la collaboration ainsi que les nouveaux concepts et les nouvelles règles métier pour former un seul système d'information. Nous parlons ici de solutions d'intégration et d'unification des systèmes d'information.

  Cette solution permet aussi de se connecter à d'autres systèmes d'information, et par conséquent cette forme joue le même rôle que la forme précédente, mais elle se situe au niveau de maturité interopérable de la collaboration.

  Les portails collaboratifs adoptants cette forme sont généralement plus adaptés et plus souples et rendent la collaboration plus efficace. Plusieurs outils pour l'entreprise adoptent cette forme. Les systèmes de gestion de contenu (CMS) adaptés au besoin métier de la collaboration présentent un exemple concret de cette solution. En conséquence, une introduction au concept de CMS est nécessaire.

Un Système de gestion de contenu (CMS) est une application flexible et fortement dynamique permettant de gérer les informations

(par exemple des documents ou des événements) et de les publier. Ces systèmes jouent ainsi un rôle d'interface entre les acteurs et le système d'information. Comme toute application, un CMS peut être sous une forme d'application web (WCMS) [88]. Cela répond en effet à plusieurs problématiques, par exemple l'accessibilité au système d'information par les acteurs de l'entreprise étant donné leur nature nomade, ou l'accessibilité des personnes non identifiées à des informations publiques, ce qui favorise la visibilité de l'entreprise. La publication des informations est faite d'une manière très simple à travers l'application en suivant un processus de publication basique où l'information passe par plusieurs états de la création à la publication.

Les WCMS présentent plusieurs avantages comme la publication rapide des informations, la gestion facile des utilisateurs, l'accessibilité au système d'information à travers un navigateur web sans installation de logiciel préalable, la facilité d'installation et de configuration [17], etc. Pour développer une application web basée sur un CMS il faut commencer par identifier les besoins de l'application (types de contenu, aspect visuel) puis développer des modules permettant ainsi au CMS d'intégrer des fonctionnalités spécifiques adaptées aux besoins métier. Ces modules restent très dépendants de l'architecture de l'application. Une fonctionnalité complexe peut demander un changement dans l'architecture principale ou l'ajout d'une nouvelle couche logicielle pour assurer l'intégrité.

Un CMS est souvent fourni avec un grand nombre de fonctionnalités de base qui reflètent une certaine culture et expérience dans le domaine de la gestion des données, par exemple un moteur de recherche ou un agenda. Quant aux workflows, ils sont communs et se limitent au processus conventionnel de création et de publication. Les outils CMS sont caractérisés par un manque d'adaptation aux processus métier. Pour adapter un CMS à un processus metier, les développeurs dans ce cas sont obligés de créer des modules pour supporter ces processus souvent complexes. Nous pensons que le fait de rajouter cet aspect processus métier collaboratif transforme un simple WCMS en un portail collaboratif adapté au métier de l'entreprise. Le besoin incessant de fonctionnalités et une évolution permanente des processus métier collaboratif d'une entreprise ajoutés à la difficulté de livrer des applications respectant les besoins métier rendent le processus classique d'adaptation des CMS au métier de l'entreprise de moins en moins efficace.

2.6 CONCLUSION

Dans ce chapitre nous avons positionné nos travaux par rapport au domaine de la modélisation des entreprises et aux systèmes d'information collaboratifs. Nous avons ainsi distingué deux courants de modélisation pour l'entreprise ; le premier dominé par une vision

métier et qui tente de structurer l'entreprise dans le but de l'analyser (modélisation des entreprises), et le deuxième dominé par une vision plus informatique, qui est monté en puissance avec l'évolution technologique, qui tente de modéliser l'entreprise dans le but de produire des logiciels permettant le bon fonctionnement de l'entreprise

Pour la modélisation des entreprises, nous avons donné un aperçu de l'évolution des méthodes de modélisation. En conséquence, nous nous sommes intéressé au métamodèle CIM-OSA pour son approche centrée sur les processus métier (la vue fonctionnelle) et le métamodèle UEML pour sa vue informationnelle. Cela nous permettra d'adopter le langage métier pour la modélisation des systémes d'information collaboratifs (SIC). Pour ces derniers nous avons défini la notion de système d'information. Nous avons vu les différentes approches (fonctionnelle, systémique et structurelle) permettant de décrire un système d'information. En nous basant sur ces derniers, nous avons donné une définition du système d'information plus générale issue de l'approche structurelle et de l'approche fonctionnelle. Cette définition met en avant la notion de processus métier que nous avons clarifié et qui décrit comment les acteurs agissent sur les informations. Dans cette définition les acteurs peuvent être des humains ou des machines. Ensuite nous avons défini la notion de collaboration et son importance pour l'évolution des entreprises. Nous avons vu aussi les différents niveaux de maturité qu'une collaboration peut atteindre. Cela a permis de clarifier la notion de SIC, et a permis de situer notre travail par rapport à l'entreprise et aux types de collaborations. En effet, un portail collaboratif peut être sous la forme d'une application web permettant aux différents acteurs de la collaboration, de manipuler les informations selon des processus métier collaboratif bien définis. Un portail collaboratif fait partie du système informatique collaboratif et il peut jouer un rôle d'interopérabilité entre les différents systèmes d'information. La collaboration dans ce cas est considérée à un niveau de maturité interopérable.

Nous avons vu, aussi, que le niveau de maturité de la collaboration et les technologies utilisées évoluent parallèlement. L'image de l'une reflète l'image de l'autre. Par extension nous pouvons passer de l'espace de description de l'un à l'espace de description de l'autre. D'un autre point de vue, un modèle du SIC peut être perçu comme une spécification d'un cahier des charges et le système informatique comme son livrable. Ainsi, une modélisation du SIC orientée métier (modélisation des entreprises) peut servir de cahier des charges pour le portail collaboratif. Cela permet de remonter en abstraction et de réduire le risque d'erreur dans la spécification des besoins métier.

Enfin, l'entreprise évoque de plus en plus le besoin de collaborer afin d'étendre son métier et gagner en puissance. D'un côté, cette collaboration évolue suivant le SIC ainsi que les changements et l'ajout des différents collaborateurs. D'un autre côté, le système d'infor-

mation doit refléter cette image variable de cette collaboration. Pour relever ce défi, l'entreprise doit avoir son autonomie sur l'évolution de son portail collaboratif. Nous pensons que le meilleur moyen, pour avoir cette autonomie, réside dans ce qu'elle sait faire, c'est-à-dire la modélisation de son métier. À partir de cette modélisation, l'entreprise doit pouvoir générer et administrer son portail collaboratif. C'est pourquoi nous proposons une modélisation des SIC basée sur la modélisation des entreprises.

3

L'INGÉNIERIE DIRIGÉE PAR LES MODÈLES (IDM)

Sommaire

Dans le chapitre précédent nous avons parlé de la modélisation des entreprises ainsi que de la modélisation des systèmes d'information collaboratifs centrée sur les processus métier. Cela nous a permis d'adopter le langage des entreprises afin de modéliser les besoins métier des portails collaboratifs. Pour produire nos portails collaboratifs, nous adoptons une approche basée sur l'Ingénierie Dirigée par les Modèles (IDM).

Le but de ce chapitre est d'introduire l'IDM et les problématiques que nous avons rencontrées pour mettre en oeuvre notre approche. Dans la section 3.1 nous donnons une définition et le principe de l'approche IDM. Cela en comparant cette approche avec d'autres approches de conception de logiciel, par exemple les approches classiques basées sur les langages technologiques. Ensuite, dans la section 3.2 nous parlons des principaux concepts sur lesquels l'IDM se base. Puis, dans la section 3.3, nous détaillons la notion de processus de production de logiciel dans une approche IDM. Dans la même section, nous donnons une classification de processus de production afin de positionner notre approche par rapport à cette classification. Enfin, dans la section 3.4, nous détaillons notre problématique associée à l'automatisation des processus de production dans une démarche IDM.

3.1 DÉFINITION ET PRINCIPE

Le génie logiciel est passé par plusieurs phases déterminantes dans le but de maîtriser la complexité grandissante du développement des logiciels. Ainsi, des avancées technologiques majeures ont vu le jour, partant des technologies procédurales jusqu'à celles des composants en passant par les technologies objets. Dans cette évolution, le modèle a toujours pris une forme passive (c'est-à-dire contemplative) en n'étant qu'un moyen d'échange d'idée lors de la phase de conception

[63]. Pour améliorer les modèles, certains se sont dit : et si le modèle devenait productif ! Et comment faire ? Et pourquoi ? L'IDM donne des réponses à ces questions.

Dès la création de la première machine programmable, l'homme a créé des langages lui permettant d'interagir avec la machine et de lui transmettre des données. Le langage machine étant une suite des zéros et des uns, un procédé permettant de passer du langage naturel au langage machine est apparu nécessaire. L'un des premiers procédés performants fut la compilation. Il s'agit d'un processus de production permettant de passer d'un langage déclaratif (sous ensemble non ambigü du langage naturel exprimant les ordres à réaliser) au langage machine [4]. Aujourd'hui, on trouve des langages, comme le C, munis de compilateurs permettant de transformer le programme d'entrée en un code exécutable par la machine. Une autre approche consiste à interpréter le code par un programme appelé « Interpréteur » qui est, souvent, écrit dans un langage compilable. Par exemple le langage Python avec son interpréteur écrit en C. Le code est alors transformé en une liste d'ordres au moment de l'exécution, mais pour des raisons de performance le résultat de cette transformation peut être stocké afin d'être rejoué si l'on exécute à nouveau le même programme.

L'apparition des langages de programmation avec leur compilateur ou leur interpréteur vise à simplifier la construction des programmes [4]. Cela a constitué une étape importante pour rendre la construction des logiciels plus accessible.

Ces dernières années, nous assistons à l'émergence d'une « nouvelle » forme d'ingénierie des logiciels, centrée sur le modèle, et appelée Ingénierie Dirigée par les Modèles (IDM) [66] [44] [64]. L'IDM offre un cadre méthodologique et technologique prometteur permettant d'unifier différentes façons de faire en un processus homogène et de favoriser l'étude des différents aspects du système.

L'IDM est une forme d'ingénierie générative qui aboutit au code du système décrit à haut niveau d'abstraction [4]. Elle se focalise sur une modélisation abstraite du système réel en un modèle, et un passage automatique de ce modèle à un langage compréhensible par la machine. Ce passage automatique, fait par la machine, rend le modèle lui même accessible par la machine et est équivalent à la compilation. En parallèle avec les approches basées sur les langages classiques, l'IDM présente une étape majeure pour rendre la construction des logiciels encore plus accessible.

Dans l'approche IDM, le modèle est au coeur du processus de développement logiciel. L'utilisation intensive et systématique des modèles dans une telle approche implique que les différents modèles soient des entités intelligibles par la machine. En parallèle avec le « tout est objet », l'IDM consiste à dire que « tout est modèle » [62].

Afin de comprendre la finalité de l'approche IDM, nous proposons de la comparer par rapport aux approches classiques basées sur les langages technologiques ainsi que l'approche naturelle. Pour cela, nous commençons par définir, selon nous, ce qu'est l'approche naturelle.

L'approche classique en V [87] d'ingénierie logicielle passe par la rédaction d'un cahier des charges, l'analyse de celui-ci par l'équipe de réalisation, la spécification détaillée, le codage, les tests (unitaires, d'intégration, de validation) puis la livraison. Les démarches agiles consistent approximativement à itérer ce cycle en le réalisant sur des échéances courtes de durées constantes. Lors de l'analyse, l'équipe identifie les concepts spécifiques du besoin, les ordonne et identifie leurs associations. Elle crée ainsi un thésaurus voire une ontologie. Puis ces concepts sont manipulés pour créer une représentation conceptuelle afin d'élaborer l'architecture ou de lister les tâches à réaliser. En l'absence de méthode ou de formalisme, l'équipe crée d'une manière ad hoc une représentation informelle du besoin, ou de l'architecture logicielle. Cela peut être qualifié d'approche naturelle basée sur une réflexion stratégique [143].

La figure 11 montre la relation entre l'approche naturelle, l'IDM et les approches classiques fondées sur les langages technologiques par rapport aux différents niveaux de conception d'une application web. Au niveau métier informel le modèle est non structuré, il se caractérise par le manque d'information. Ce modèle est souvent peu organisé et peu détaillé, ce qui pousse à dire qu'il traduit une vision restreinte du problème. Le niveau métier formel permet d'avoir un modèle plus complet et plus formel. Ce modèle garde la même sémantique que le premier en rajoutant d'autres concepts et relations. Le niveau technique permet d'avoir un modèle exprimant la solution aux besoins métier en utilisant un langage employé pour une finalité technique (c'est-à-dire Informatique) par exemple UML. Dans ce modèle des informations sur le type de la plateforme et sur l'architecture de l'implémentation sont présentes. Finalement, les niveaux technologiques de haut niveau et celui de bas niveau couvrent les différentes technologies utilisées pour produire l'application.

À la différence des approches classiques fondées sur les langages technologiques, l'IDM permet de couvrir tous les niveaux de conception d'une application web. Les techniques de passage d'un niveau à un autre, quant à elles, restent communes entre les différentes approches. Enfin, les techniques de passage entre les différentes approches se limitent à la transcription (ou génération de code).

Dans une approche IDM, il faut faire la distinction entre la façon dont on exprime les besoins (ici les exigences métier), le but de cette expression (ici le code de l'application) et enfin la méthodologie adoptée pour passer de l'un à l'autre (ici le processus de production). L'expression des besoins ne doit pas être polluée par les techniques de

passage de celles-ci au code final et doit se formuler dans un langage métier situé à un haut niveau d'abstraction.

La formalisation des besoins métier (c'est-à-dire le passage du niveau informel au niveau formalisé), la virtualisation des besoins métier (c'est-à-dire le passage du niveau formalisé au niveau technique) et enfin la production du code de l'application (c'est-à-dire le passage du niveau technique aux niveaux technologiques) peuvent être faites d'une manière automatique par transformations de modèles [64]. Ceci entraîne une succession de transformations de modèles, chaque transformation permettant de faire passer d'un niveau d'abstraction au niveau d'abstraction inférieur, jusqu'à atteindre le niveau voulu. L'enchaînement et l'organisation de ces transformations et très important et constitue un processus que nous appelons *processus de production de logiciels*.

Si le processus de production logiciel n'est pas automatisé, il est alors réalisé par intervention d'un spécialiste : expert, analyste, développeur, intégrateur, etc. On comprend alors la force de l'IDM si l'expression du cahier des charges se fait par un modèle pouvant être automatiquement transformé et si en plus ce cahier des charges utilise un modèle exprimé par un DSL spécifique au métier ciblé permettant au client de lui même exprimer son besoin sans intermédiaire.

Dans ce qui suit, nous présentons les principaux concepts de l'IDM.

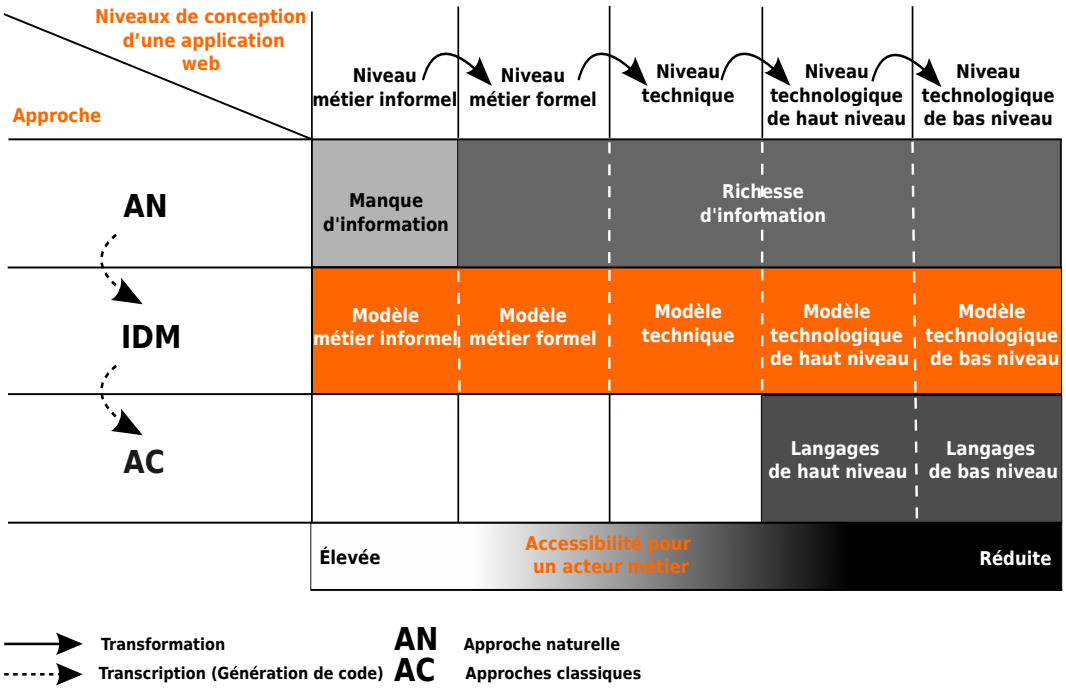


FIGURE 11: Niveaux de modélisation

3.2 LES PRINCIPAUX CONCEPTS DE L'IDM

Dans l'approche IDM, nous pouvons identifier les modèles et les transformations de modèles. Les transformations de modèle sont aussi des modèles. Ce qui confirme que dans une approche IDM tout est modèle. Les transformations de modèle sont souvent accompagniées par un moteur de transformations permettant leur exécution. Les transformations de modèles permettent de rendre les modèles intelligibles par la machine.

Dans cette section nous détaillons les différents types de modèle et de transforaiton de modèle.

3.2.1 *Les modèles*

Selon la décomposition de l'OMG [100] nous pouvons trouver trois types de modèles, comme illustré à la figure 12 ;

- Le **modèle** qui peut être défini comme un ensemble d'informations ou de connaissances en relation avec un domaine donné. Cet ensemble représente un point de vue d'un système réel à un niveau abstrait. Un modèle peut être vu comme une simplification du système réel permettant de répondre à toutes les questions sous-jacentes à un point de vue donné du système [66]. Par exemple un modèle BPMN (Business Process Model and Notation) est un modèle qui permet de décrire les processus métier.
- Le **métamodèle** est un modèle permettant de décrire le modèle. On dit, dans ce cas, qu'un modèle est conforme à un métamodèle. En parallèle avec la notion de langage informatique, le métamodèle est la grammaire et le modèle est le programme. Par exemple l'ensemble des définitions des concepts de BPMN forme un métamodèle.
- Le **métamétamodèle** est un modèle permettant de décrire les métamodèles. C'est donc aussi un métamodèle. Afin d'éviter d'avoir un autre métamodèle pour décrire le métamétamodèle, ce dernier se décrit avec lui-même. Dans le parallèle avec la notion de langage, le métamétamodèle est au métamodèle ce que la description BNF (Backus Naur Form) est à une grammaire, le métamodèle est au modèle ce que la grammaire est à un langage. Originellement, l'OMG a défini le langage MOF comme métamétamodèle [102]. Cependant, Eclipse [36], et notamment son extension EMF [28] ont défini le langage ECore [28], qui s'impose comme un standard de facto. Ce langage est actuellement pris en compte au niveau de l'OMG.

Plusieurs outils de modélisation existent, nous citons par exemple le projet EMF de l'environnement Eclipse ayant ECore comme métamétamodèle. EMF propose un ensemble de fonctionnalités permettant de créer ainsi qu'instancier son propre métamodèle.

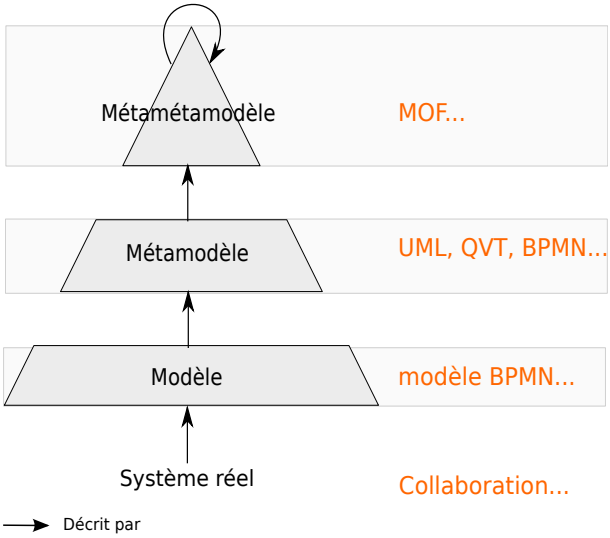


FIGURE 12: Classification des modèles (OMG)

3.2.2 *Les transformations de modèles*

Les transformations de modèles permettent de décrire comment modifier ou construire d'autres modèles en se basant sur un ensemble de modèles d'entrée. Cette description est faite à travers des règles. En IDM les transformations de modèles peuvent être des transformations de modèle vers modèles ou des générations de code ;

- La **transformation de modèle vers modèles** permet, d'une manière générale, de passer d'un espace de modélisation à un autre selon des règles de transformation. Ces règles décrivent comment les concepts des métamodèles sources se traduisent en concepts des métamodèles de destination. Plusieurs langages de transformation de modèle existent, comme QVTO [38], ATL [35] et QVT [105] qui est le standard de l'OMG. La figure 13 illustre la relation entre les transformations de modèle avec les différents modèles (modèle, métamodèle et métamétamodèle). Sur cette figure, une transformation est conforme à un langage de transformation qui lui est conforme au métamétamodèle. Cette transformation est typée par les différents métamodèle sur lesquels cette transformation s'applique. Les modèles d'entrées et de sorties doivent être conformes à ces métamodèles.
- La **génération de code** est une transformation de modèle dont la destination est du texte. Elle est aussi appelée transformation de modèles vers texte. Il n'y a aucune restriction sur le texte généré, qui peut être aussi bien du XMI, du Java ou du Python. Plusieurs langages de génération de code existent, comme Acceleo [34] ou le standard OMG MOF2Text [99]. La figure 14 illustre la relation entre les générations de code avec les différents modèles (modèle, métamodèle et métamétamodèle). Sur cette figure, une généra-

tion de code est conforme à un langage de génération qui lui est conforme au métamétamodèle. Cette génération de code est typée par les différents métamodèle sur lesquels cette génération s'applique. À la différence des transformations de modèles vers modèles, une génération de code ne peut avoir que des modèles d'entrées. Ces modèles doivent être conformes aux métamodèles typant la génération de code. En sortie, la génération de code produit du texte qui est sauvegardé dans des fichiers à leur tour sauvegardés dans des dossiers.

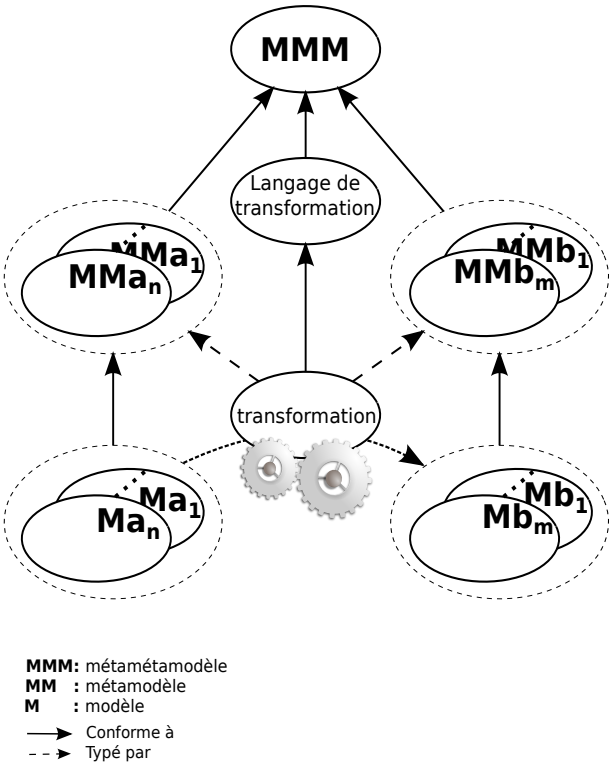


FIGURE 13: Relation entre les transformations de modèles et les modèles (modèle, métamodèle et métamétamodèle)

Selon les niveaux de conception d'une application web illustrés sur la figure 11, nous pouvons regrouper les différentes transformations d'un processus de production de logiciels en trois groupes. Chaque groupe permet de réaliser une phase de conception de l'application à développer. Un groupe correspond à une ou plusieurs transformations dont le but est de passer à la phase suivante jusqu'à la génération de code. Ces groupes sont les suivants :

– **métier informel vers métier formel** (ou modélisation des exigences [64]) : Cette phase joue le rôle d'assistant de modélisation permettant d'orienter le concepteur vers un modèle plus complet et plus formel de ses besoins. C'est l'équivalent d'un expert métier en phase d'identification des exigences. L'identification des exigences nécessite souvent une forte interactivité entre l'expert métier et le client. Par extension, l'interactivité entre le processus

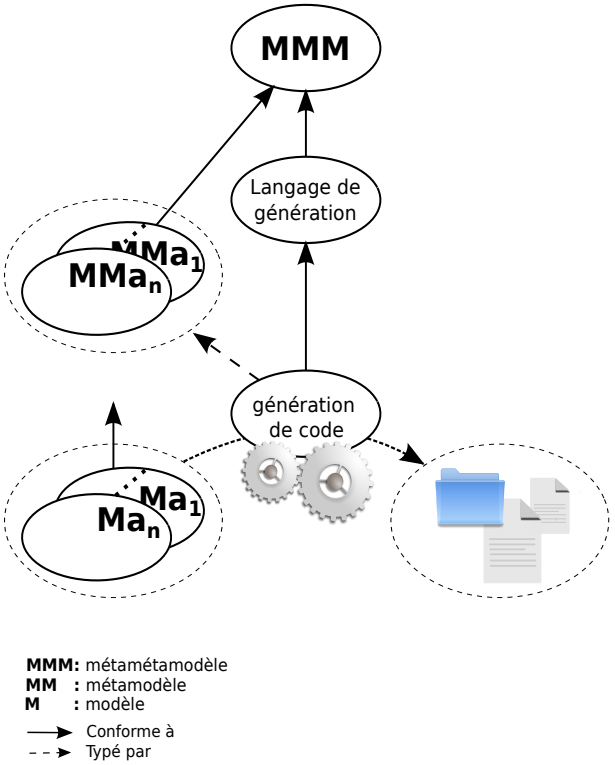


FIGURE 14: Relation entre les générations de code et les modèles (modèle, métamodèle et métamétamodèle)

de production de logiciels et le concepteur devient une nécessité. Cette phase nécessite le plus d'intelligence dans les transformations afin d'analyser, de détecter et corriger les erreurs et d'enrichir le modèle informel initial. L'identification des processus métier de la collaboration est un exemple de cette phase. Selon le formalisme utilisé pour modéliser ces processus métier le modèle d'exécution de ces processus peut changer. Dans cette phase, la description concrète du modèle d'exécution des processus métier n'est pas utile. En effet, les métamodèles décrivant les processus métier sont accompagnés d'une description implicite de leur modèle d'exécution. Ce modèle d'exécution, selon nous, peut être décrit dans un modèle indépendant qui peut servir comme modèle d'entrée d'une transformation de modèle.

Un modèle d'exécution des processus métier décrit comment un processus métier s'exécute.

– **Métier formel vers technique** : Cette phase joue le rôle d'architecte d'application permettant de traduire les besoins métier en une architecture technique de l'application. Les détails de l'implémentation, par exemple les classes et les méthodes, sont ainsi ajoutés au modèle. La description du modèle d'exécution des processus métier et les différentes classes impliquées sont un exemple de cette phase. Dans ce cas, les différents concepts sont ajoutés d'une manière indépendante du modèle métier. Ce modèle peut être construit par une transformation ou préconstruit

et introduit aux différentes transformations comme un modèle d'entrée.

– **Technique vers technologique** : Cette phase joue le rôle du développeur permettant de traduire les choix techniques en détail technologiques. Le choix des bibliothèques, qui seront utilisées lors du développement, est ainsi décrit. Une implémentation du module permettant l'exécution des processus est un exemple de cette phase. Ce module peut être considéré comme une bibliothèque indépendante et réutilisable au niveau technologique.

3.3 PROCESSUS DE PRODUCTION DE LOGICIELS

Dans notre approche nous nous intéressons à la compilation de modèle dans une approche IDM. La compilation est concrétisée par la notion de processus de production de logiciel.

Généralement, sur un plan technique, un informaticien a besoin d'un cahier des charges complet décrivant le système à réaliser et d'un ensemble de connaissances sur des langages lui permettant de réaliser le système. On peut voir l'informaticien comme quelqu'un ayant pour rôle de transformer le cahier des charges écrit en langage humain en un langage compréhensible par la machine. Cela en se basant sur les connaissances acquises précédemment (grammaire, syntaxe, bibliothèque, etc.). Cette opération de transformation suit généralement un processus de production de logiciels. Ce processus est décrit par les différentes activités à appliquer sur le cahier des charges ainsi que le flux d'exécution de celles-ci orienté selon un certain nombre de contraintes, par exemple le choix des bibliothèques les plus adaptées ou le type de la plate-forme cible. Selon nous, dans une approche IDM, les activités du processus de production peuvent être totalement automatisées et sont centrées sur le modèle. Généralement, dans une telle approche, les activités sont de différentes natures, par exemple des transformations de modèles vers modèles ou des générations de code.

Un processus de production permet d'ordonner les activités afin d'être exécuté par le système. D'une manière générale, les processus de production prennent un ensemble de connaissances, qui peut être vide, en entrée à un niveau d'abstraction de haut niveau, par exemple un langage naturel, ou de bas niveau, par exemple le langage Python, afin de produire un ensemble de connaissances situées à un même ou un autre niveau d'abstraction.

D'un autre point de vue, un processus de production peut être vu comme une transformation généraliste (c'est-à-dire un langage de transformation spécifique au domaine, voir section 6.6) visant à mettre au premier plan les activités du processus de production et les décisions permettant de passer d'une activité à une autre de ce processus. Ces activités sont vues comme des composants réutilisables [145]

ne contenant pas les décisions clefs propres au processus de production. Dans notre contexte un processus de production nous permet de passer d'un modèle MACoP au code du portail collaboratif.

Selon le type des modèles d'entrée et la finalité du processus de production, nous pouvons identifier trois catégories de processus production :

- **Processus de production pour la rénovation/migration** : Ce type de processus permet de passer d'une application écrite dans un langage technologique en une application qui a les mêmes fonctionnalités écrites dans un autre langage technologique. Le but principal est la rénovation ou la migration de l'application existante. Nous citons dans cette catégorie le projet Blu Age [12]. Afin de rénover l'application complète, les bibliothèques utilisées dans l'application source doivent être rénovées automatiquement ou manuellement pour l'application cible.
- **Processus de production pour l'implémentation à partir du niveau technique** : Le but de ces processus est de simplifier le travail du développeur en lui proposant un métamodèle lui permettant de décrire l'application au niveau technique là où les exigences métier sont décrites d'une manière implicite. Ce métamodèle est indépendant des technologies à utiliser pour l'implémentation de l'application, par exemple UML ou WebML [132].
- **Processus de production pour l'implémentation à partir du niveau métier** : Le but de ces processus est de passer d'une description des exigences métier en une application finale. Dans ce cas, c'est le travail de l'architecte de l'application est automatisé. Ce type de processus présente un défi [67] puisqu'il requiert des transformations intelligentes et des prises de décision sur les détails de l'implémentation. Ces processus doivent être accompagnés de puissantes transformations d'optimisation afin de produire une application rapide et robuste. Les solutions BPM (Business Process Management) comme BonitaSoft [18] présentent une tentative timide de ce type de processus.

Une des étapes les plus importantes dans un processus de production de logiciel est la génération de code. Celle-ci peut être faite à partir du modèle technique sans passer par un modèle technologique. Une deuxième approche consiste à passer par un modèle technologique.

La génération de code, dans ce cas, joue le rôle d'une simple dictée en transcrivant directement le modèle technologique (*pretty printing*). Cette approche permet d'appliquer des transformations d'optimisation sur le modèle technologique, une chose que la première approche ne peut faire.

Dans notre cas, nous visons l'automatisation complète du processus de production. De ce fait, le processus est, complètement, exécuté par le système. Ce n'est pas le cas pour les processus de production

semi-automatiques qui impliquent une collaboration entre le système et des informaticiens.

Pour la production de nos portails collaboratifs, nous avons besoin d'un outil permettant de décrire nos processus avec ce qu'ils impliquent d'interaction avec le concepteur ainsi qu'un moteur d'exécution permettant de les exécuter.

3.4 NOTRE PROBLÉMATIQUE

Dans le cadre d'une approche IDM [66] les processus de production sont généralement automatisés et centrés sur le modèle. Par conséquent, le modèle est considéré comme un objet auquel nous pouvons poser des questions (c'est-à-dire faire des opérations) et non comme un simple fichier. Les activités sont principalement des transformations de modèles vers modèles, de la génération de code ou des boîtes noires permettant d'exécuter un code quelconque comme de la vérification. Les contraintes permettant d'orienter le flux d'exécution, quant à elles, portent sur les modèles manipulés par le processus. Cela constitue une spécificité pour les processus de production IDM par rapport aux workflows qui présentent une vue automatisable des processus métier [45].

D'une manière générale, un processus de production de logiciels prend un ensemble de modèles décrivant l'application à concevoir et produit l'ensemble de code source correspondant. Au cours de l'exécution du processus, le concepteur est amené à faire des choix, comme par exemple spécifier l'emplacement des modèles d'entrées ou la destination du code à générer. Ces choix peuvent être fixés à l'avance, mais il peut être intéressant de rendre certains d'entre eux interactifs : le système demande alors les valeurs lors de l'exécution.

Il arrive souvent que des processus de production de logiciels aient des parties similaires. Il est intéressant de pouvoir capitaliser sur ces parties, en les réutilisant dans différents processus de production. Une approche intéressante est de voir le processus de production comme étant une activité elle-même réutilisable dans un autre processus.

Dans un contexte où les technologies de transformation de modèle et de génération de code sont devenues de plus en plus nombreuses, un processus de production doit s'abstraire par rapport à la technologie de transformation afin de permettre d'utiliser différentes technologies dans un même processus.

En résumé, selon nous, un processus de production IDM doit respecter les points suivants :

- Le processus de production doit décrire un enchaînement d'activités avec un flux d'exécution orienté selon des contraintes (c'est-à-dire, une exécution conditionnelle).

- Dans un processus de production, les modèles doivent être manipulables (c'est-à-dire pouvoir poser des questions au modèle).
- Un processus de production peut être interactif. Une gestion simple des interactions est alors nécessaire, tant avec l'utilisateur qu'avec le système de fichier.
- Dans un processus production, les activités doivent être réutilisables afin de simplifier la modélisation.
- Le processus de production peut être hétérogène et indépendant des technologies utilisées par les activités. Nous devons avoir, dans ce cas, la possibilité d'ajouter facilement des activités de différentes technologies.

Un processus de production de logiciels est avant tout un procédé logiciel [60]. Citons comme exemple de procédé logiciel SPEM [106] ou SPEM4MDE [121]. Les activités d'un processus de production logiciel sont orientées vers le développement de l'application et non la gestion des équipes de développement comme c'est le cas pour les procédés logiciels. En effet, les procédés logiciels sont des approches généralistes centrés sur l'organisation du travail au sein d'une équipe dont le but est de produire des logiciels.

Le problème d'enchaînement des transformations, dans une approche IDM, a été traité de différentes manières. Nous trouvons, par exemple, les approches classiques qui se basent sur des langages de script comme ANT [93]. Ces approches ne sont pas dédiées aux approches IDM et restent difficiles à adapter. Dans [9], les auteurs présentent l'environnement MCC permettant la description textuelle de l'enchainement des transformations de modèle. Cette approche permet la réutilisation des transformations ainsi que l'exécution conditionnelle de leur enchainement. Tout en adoptant une description textuelle des processus, le moteur MWE [37] permet l'enchainement simple (non conditionnel) des activités et ne se limite pas seulement aux transformations de modèles. D'autres approches ont adopté une description à base de modèle ce qui a permis une utilisation plus simple. Nous citons par exemple le moteur Acceleo [34] qui permet l'enchaînement simple des templates de génération de code Acceleo ou l'outil ATLFlow [113] qui permet de décrire un enchainement conditionnel des transformations ATL. Ces outils sont dépendants de la technologie utilisée pour les activités. Le besoin d'enchainer des activités avec différentes technologies a été évoqué dans plusieurs approches. Nous citons, par exemple, transML [53] qui permet un enchainement simple des transformations ou le travail fait dans [54] permettant un enchainement conditionnel de différentes activités. Dans [145] les auteurs présentent l'outil UniTI permettant un enchainement simple des processus de production. Cet outil se focalise sur la réutilisation des activités de différentes technologies. Les technologies, dans ce cas, sont décrites dans le métamodèle utilisé par cette ap-

proche. Cela la rend dépendante des technologies utilisées pour les activités.

Ces approches montrent des manques au niveau de l'interactivité avec l'utilisateur et avec le système de fichier. La plupart de ces approches sont dépendantes des technologies des activités, par exemple le moteur Acceleo ou l'outil ATLFlow. Les approches revendiquant l'hétérogénéité des technologies présentent une dépendance aux technologies au niveau du métamodèle, par exemple UniTI. Cela rend l'ajout d'une nouvelle technologie difficile. Aucune de ces approches ne considère le modèle comme un objet manipulable. Les conditions permettant d'orienter le flux d'exécution portent généralement sur des expressions manipulant des variables primitives. Enfin, ces approches restent difficiles à adapter pour respecter les points cités ci-dessus.

3.5 CONCLUSION

Dans cette section nous avons défini l'approche IDM ainsi que les concepts sur lesquels elle se base. Les différents travaux que nous avons présentés nous mènent vers la conclusion que l'IDM n'est pas une nouveauté dans le monde de l'informatique, c'est un concept naturel déjà existant et en constante évolution. Aujourd'hui l'IDM tente de remonter, encore, en abstraction. Le but principal est de rendre les langages de bas niveau accessibles principalement par la machine et faire apparaître des langages de plus en plus proches du langage humain, facilement compréhensibles et exprimant les besoins métier de l'utilisateur. Les différentes approches de passage d'un niveau à un autre (c'est-à-dire la compilation ou l'interprétation), quant à elles, n'ont pas changées. Nous trouvons ainsi la compilation de modèle sous forme de processus de production et l'interprétation de modèle par un interpréteur.

Nous avons également défini la notion de processus de production de logiciels que nous pensons utile pour l'automatisation de l'enchainement de transformations complexes et interactives par exemple les transformations de formalisation des modèles métier informel ou les transformations d'optimisation au niveau technologique. Cette définition nous a permis d'identifier les points qu'un outil de modélisation et d'exécution de processus de production doit respecter.

Si des outils respectant certains de ces points ont été développés (par exemple UniTI [145] ou ATLFlow [113]), aucun d'entre eux ne répondait parfaitement à notre besoin. Nous avons donc développé notre propre outil baptisé **PMS+** (Process Manufacturing Software and +).

4

INGÉNIERIE DES APPLICATIONS WEB

Sommaire

Dans les chapitres précédents, nous avons introduit la modélisation des systèmes d'information ainsi que celle des entreprises afin d'adopter leurs langages pour modéliser nos portails collaboratifs. À partir de cette modélisation, nous produisons nos portails collaboratifs, opérationnels, en suivant une approche IDM (Ingénierie Dirigée par les Modèles). Pour cela nous avons défini l'approche IDM.

Nos portails collaboratifs sont des applications web centrées sur les processus métier. Nous allons dans ce chapitre étudier les approches basées sur la modélisation des applications web.

Dans la section 4.1 de ce chapitre, nous parlons de l'historique du développement web et de ses problématiques. Ensuite, dans la section 4.2 nous positionnons les portails collaboratifs selon une classification des applications web. Cette classification est centrée sur les critères de l'interactivité, entre l'utilisateur et l'application, ainsi que sur l'utilité de l'application. Puis, section 4.3, nous parlons des différents courants de modélisation des applications web comme solution pour simplifier leurs développements. Enfin, dans la section 4.4, nous détaillons l'approche WebML, considérée comme une des approches les plus importantes, afin de clarifier les limites des approches de modélisation des applications web.

4.1 HISTORIQUE DU DÉVELOPPEMENT WEB

La première application (http://first-website.web.cern.ch/) est apparue en 1989 (http://info.cern.ch/) dans le laboratoire du CERN (Conseil Européen pour la Recherche Nucléaire)(http://home.web.cern.ch/). Cette application a été proposée par Tim Berners-Lee pour aider les physiciens et les ingénieurs à partager leurs informations.

Les années 90 ont été marquées par l'apparition des premières applications web. Avec l'évolution technologique et la multiplication des technologies dans ce domaine, les applications web sont devenues de plus en plus riches au niveau du rendu et complexes au niveau de l'implémentation. Cette évolution reflète l'évolution technologique. Selon [139], les applications web ont vu cinq générations : la première génération porte sur les applications simples basées sur un contenu textuel (navigateurs textuels) où l'apparence est sommaire ; la deuxième génération a introduit l'utilisation des technologies CGI (Common Gateway Interface) permettant ainsi la communication des données dans les deux sens (client-serveur), mais qui reste toujours aussi simple que la première génération au niveau de la présentation du contenu ; la troisième génération, avec l'apparition des navigateurs web avancés, portait sur la présentation des données (mise en page, style, etc.) de l'application plus que sur le contenu ; la quatrième génération, avec la montée en puissance des ordinateurs, portait sur l'introduction des nouvelles fonctionnalités, par exemple l'introduction des nouveaux types de contenus média, comme le contenu vidéo, et l'utilisation des nouveaux langages de programmation afin d'augmenter l'interactivité, par exemple Java ; actuellement, la cinquième génération porte sur les besoins des développeurs au même titre que les besoins de l'utilisateur. Les besoins de l'utilisateur concernent les fonctionnalités et les informations présentes dans l'application. Les besoins des développeurs, quant à eux, sont concrétisés en proposant des environnements et des API facilitant le développement des applications web ainsi que leur maintenance. La particularité technique des applications web appartenant aux nouvelles générations se caractérise, généralement, par la répartition de l'application à la fois du côté client et du côté serveur. Cette répartition rend les applications web difficiles à concevoir. En effet, les applications Web, par leur sensibilité sécuritaire, leur architecture technique (répartie sur le client et sur le serveur) et leur richesse en termes d'interaction sont différentes des logiciels et applications informatiques classiques [119].

Étant donné la simplicité des applications de première génération, le développement de ces dernières a pris des formes *ad hoc* qui sont devenues inadaptées pour le développement des applications de nouvelles générations. Les applications, de nouvelles générations, requièrent de plus en plus de technologies pour fonctionner, tant côté client (HTML, CSS, JavaScript, etc.) que côté serveur (Python, Ruby, Java, J2EE, MySQL, etc.).

De plus, ces applications peuvent être basées sur différentes architectures, par exemple MVC (Modèle-vue-contrôleur), trois- tiers, etc. Pour réduire la complexité du développement et la mise en œuvre des applications web, une réflexion profonde sur les techniques et

méthodes de développement de ces applications a vu le jour. Une telle complexité doit être gérée d'une manière structurée afin d'éviter les risques d'erreurs et de non-aboutissement du projet. Ainsi en 1998 l'ingénierie des applications web a pris forme et a été reconnue comme une discipline à part entière [16]. Selon [52], l'ingénierie des applications web est la combinaison de l'utilisation des principes scientifiques, d'ingénierie, de principes de gestion et d'approches systématiques dans le but de réussir à développer, déployer et maintenir à haut niveau de qualité des systèmes et des applications basés sur le Web. Cette discipline porte sur plusieurs thèmes, allant de la modélisation des exigences au test, en passant par le design de ces applications. Aujourd'hui, on trouve plusieurs frameworks pour le développement des applications web, comme Struts [133], Symphony [135] ou Django [31], dont un des objectifs est de simplifier le développement web en proposant une décomposition logique de l'application. Selon nous, cela reste insuffisant et ne fait que résoudre en partie le travail du développeur. En effet, en plus de l'apprentissage des différents langages impliqués dans le développement de l'application, il faut maîtriser les différentes API mises à disposition dans le but de simplifier le développement. Ces API présentent une abstraction ascendante (c'est-à-dire de la technologie aux concepts abstraits) permettant de capter des besoins technologiques et techniques afin de les réutiliser pour, enfin, former un modèle technologique.

Le design de l'application web est l'aspect esthétique de l'application (style, interaction, etc.)

La nature variable et évolutive des besoins métier, rend le développement des applications web de plus en plus complexe. Avec l'apparition des applications internet riches, cette complexité a atteint un niveau équivalent à celui des applications classiques.

Une réduction de cette complexité est nécessaire afin de produire des applications web respectant les besoins métier. Selon nous, une amélioration de la communication entre les acteurs métier et les informaticiens est nécessaire pour éviter les erreurs qu'une application peut rencontrer. Une mauvaise communication engendre une mauvaise application qui peut avoir des répercussions graves sur le bon déroulement du métier de l'entreprise. Dans ce sens, le Cutter Consortium [25] a mentionné un nombre important de problèmes rencontrés dans le développement des applications Web.

- Seulement 16% des applications répondent aux besoins métier réels,
- Les délais de livraison des applications ne sont pas respectés dans 79% des cas,
- Les budgets consacrés au développement des applications sont dépassés dans 63% des cas,
- La pauvreté des spécifications fonctionnelles touche 53% des cas,
- La qualité des livrables est mauvaise dans 52%.

Dans le chapitre 2 nous avons parlé de la nécessité de l'entreprise de gérer son portail collaboratif en toute autonomie. Cela permet à

l'entreprise de devenir plus réactive et de satisfaire ses contraintes dans un délai réduit. Afin de satisfaire ce besoin ainsi que les problématiques citées ci-dessus, nous pensons qu'une démarche basée sur la modélisation des systèmes d'information collaboratifs afin de générer le code complet nous parait la plus intéressante à explorer. Avec une telle approche, le problème de communication peut être réduit et ainsi générer des portails collaboratifs respectant le métier de l'entreprise.

4.2 CLASSIFICATION DES APPLICATIONS WEB

Afin de positionner les portails collaboratifs dans le domaine de l'ingénierie des applications web, nous allons tenter de donner une classification des applications web.

Ces applications peuvent être classifiées selon plusieurs critères : technologie utilisée, architecture, supports, utilité, etc.

Notre travail portant sur les applications collaboratives, nous nous intéressons à l'aspect interactif entre l'application et l'utilisateur. Ainsi, nous distinguons deux types d'application : les applications de diffusion, caractérisées par une faible interactivité, et les applications de collaboration avec une forte interactivité.

Parmi les autres critères nous retenons, aussi, la classification portant sur l'utilité des applications web faite dans [50]. Dans cette classification nous distinguons quatre catégories : les sites web de présence sur le Web et les sites web « catalogues » permettant essentiellement de diffisuer les données ; et les sites web orientés services ainsi que les systèmes d'information basés sur le Web (SIW) caractérisés par leur richesse en fonctionnalités permettant d'agir sur les données.

Un service est ensemble de fonctionnalités agissant sur les données du site web. Un SIW est un système d'information dont le fonctionnement est assuré par des applications web.

Dans ce qui suit, nous détaillons notre classification. Pour chaque catégorie de cette classification, nous détaillons les catégories d'utilités correspondantes.

4.2.1 *Les applications web de diffusion (Web 1.0)*

Le web a commencé par une présentation statique de l'information sous forme de pages Web reliées par des liens hypertextes. Ces pages étaient écrites par les concepteurs informaticiens, puis déployées sur le serveur pour être enfin interrogées par l'utilisateur. Pour modifier le contenu et la structure de ces pages, les concepteurs informaticiens modifiaient directement le code. Afin d'éviter de modifier le code, les concepteurs informaticiens ont mis en place un système dynamique qui permet de prendre en compte l'évolution des bases de données. Cependant, les interfaces produites avec ces applications ne permettent pas à l'utilisateur final de l'application d'appliquer des actions métier comme la modification ou la suppression du contenu, ce qui les rend faiblement interactives et limitées à la diffusion de l'informa-

tion. Ce type de site web est souvent appelé le Web1.0. En parallèle avec la classification faite dans [50], concernant l'utilité, nous pouvons distinguer, dans la catégorie des applications web de diffusion, deux types d'applications web :

- les sites web de présence sur le Web qui ont comme objectif la mise en ligne d'informations, à des fins publicitaires ou éducatives par exemple.
- les sites web "catalogues" ou à forte densité de données qui sont caractérisées par le fait qu'ils publient une masse importante de données, selon une structure hypertexte complexe, mais pauvre en services.

4.2.2 *Le web collaboratif (Web 2.0 et plus)*

Le besoin de collaboration et de travail à distance devenant de plus en plus important, les applications du Web 1.0 sont devenues inadaptées aux besoins de collaboration. Le passage à un autre type d'application est devenu nécessaire. Ce type d'application est identifié comme le Web 2.0. Cette classe d'application offre à l'utilisateur des interfaces de contrôle lui permettant d'ajouter, modifier ou supprimer du contenu. Avec l'évolution technologique (par exemple AJAX) les applications WEB 2.0 s'approchent de plus en plus des applications bureau classique pour devenir plus dynamiques et plus interactives avec un excellent rendu de l'interface utilisateur. Le traitement des données a été amélioré et migré vers le navigateur web. Cela permet d'éviter le chargement systématique des pages à chaque action de l'utilisateur. Des micros synchronisations avec le serveur sont faites afin d'assurer la cohérence des données. Ces applications sont identifiées comme les applications web riches. D'autres types d'applications web ont vu le jour afin de s'adapter aux nouvelles technologies. Par exemple les applications web mobiles adaptées aux technologies mobiles comme les téléphones portables. Ces applications permettent d'avoir un rendu plus ergonomique et adapté aux contraintes des technologies mobiles (taille de l'écran, technologie tactile, etc.).

Une application bureau classique est une application accessible localement à partir de la machine sur laquelle elle est installée.

Afin d'améliorer l'exploitation et l'exploration des données présentes sur le web, les applications web sémantiques [128] proposent de rendre toutes les données (transactions, documents, etc.) présentes sur le web accessible par la machine. Ce qui n'est pas le cas pour les applications web classiques où la majorité des données ne sont compréhensibles que par l'humain. Cela sert, par exemple, pour la recherche des données sur le web. Les résultats d'une recherche, dans ce cas, sont plus pertinents.

Le web collaboratif est un terme applicable sur toute application web permettant de collaborer autour de processus bien définis. Que le type de l'application web soit riche ou mobile, sémantique ou classique, ne fait qu'augmenter et améliorer les possibilités d'interaction,

entre l'application web et l'utilisateur, et rend l'application web plus intelligente et plus contextualisée. Le type de l'application web ainsi que les technologies utilisées pour la mettre en oeuvre dépendent en premier des besoins métier de la collaboration et du type d'interaction voulue (tactile, vocale, intelligence ambiante, monde virtuel, rendu en 3D, etc.).

En parallèle avec la classification faite dans [50], concernant l'utilité, nous pouvons distinguer, dans la catégorie web collaboratif, deux types d'application web :

- les sites web orientés services destinés à offrir des services spécifiques comme les applications de messagerie électronique. Ces sites reposent souvent sur des bases de données de grande taille avec une structure simple. Ils se focalisent davantage sur les services fournis exécutés à travers des processus souvent courts (qui ne s'étendent pas sur un long laps de temps).
- les systèmes d'information basés sur le Web (SIW) qui combinent la mise à disposition et la gestion de données complexes avec des services interactifs orchestrés par des processus métier complexes couvrant les différents aspects de la collaboration. Les portails collaboratifs entrent dans cette sous catégorie.

Enfin, un portail collaboratif basé sur le web, comme son nom l'indique, est une application appartenant au web collaboratif ayant un niveau de complexité élevé des données et des processus.

4.3 MODÉLISATION DES APPLICATIONS WEB

La modélisation des applications web, comme la modélisation des applications classiques, a pour but de simplifier la complexité et de faciliter la transition de la phase d'analyse à la phase d'implémentation. Cette modélisation comporte différentes vues afin de couvrir tous les aspects d'un même système. Deux courants de modélisation des applications web peuvent être identifiés : la modélisation majoritairement centrée sur les aspects informatiques ou technologiques de l'application (vision informatique) considérée comme la réponse aux besoins métier ; et la modélisation majoritairement centrée sur les besoins métier eux-mêmes (vision métier). Nous allons citer quelques approches existantes en les classant selon ces deux courants de modélisation : la vision informatique (**AVI**) puis la vision métier (**AVM**). Beaucoup d'approches se classant dans la vision informatique, nous proposons une synthèse de ces approches.

4.3.1 *Une vision informatique*

Un système hypertexte est un ensemble de noeuds, par exemple des pages web, reliées par des arcs. Dans un tel système, les arcs sont des liens hypertextes permettant la navigation entre les noeuds du système.

Les approches adoptant une vision informatique modélisent l'application comme un système hypertexte. Cela en se basant sur ce qui est appelé communément le modèle de navigation. Sachant qu'une

application web est une solution à un besoin métier donné. Ce type de modélisation reste une modélisation de la solution et non du besoin.

Le terme hypertexte a été utilisé au milieu des années 60. Ensuite, plusieurs applications ont concrétisé ce terme afin de former des systèmes hypertextes. Mais il n'a pris de l'ampleur qu'avec l'apparition du web. Avec cette apparition, les premières méthodes de conception des applications à base d'hypertexte ont vu le jour. HDM [90] fut la première à proposer une approche structurelle inspirée du modèle Entité/Association (EA).

authoring in-the-large est le fait de décrire l'application à gros grains. Les détails, comme le code d'une opération, n'en font pas partie.

HDM se focalise sur le *authoring in-the-large*, c'est-à-dire la description des noeuds ainsi que les liens entre les noeuds qui forment un système hypertexte. Le contenu des noeuds n'est pas pris en compte. Cela afin de garder une modélisation indépendante des technologies utilisées pour décrire les noeuds.

HDM propose une approche inspirée du modèle Entité/Association (EA) pour présenter les systèmes hypertextes. Dans cette approche les entités prennent une autre définition. Une entité (définie au niveau type) dans HDM appartient au domaine métier, par exemple une facture est une entité. Les entités sont composées par des composants ordonnés permettant de décrire la structure d'une entité, par exemple la liste des produits associée à la facture. Un composant peut être composé d'autres composants ou d'unités. Une unité présente l'élément atomique dans HDM, par exemple un texte décrivant la facture. Le contenu des unités n'est pas à spécifier dans le modèle HDM et est considéré comme faisant partie de l'*authoring in-the-small*. Cette modélisation permet de décrire une entité sous une forme hiérarchique. Dans HDM, une entité peut avoir plusieurs perspectives, par exemple la facture en français ou en anglais. Si une entité à plusieurs perspectives alors tous les composants la formant ont aussi les mêmes perspectives.

authoring in-the-small est le fait de décrire l'application à fins grains. Ce sont les détails de l'application, par exemple le code d'une opération.

HDM propose plusieurs types de liens ; les liens des perspectives permettant de relier les unités d'une même perspective ; les liens structurels permettant de naviguer entre les composants d'une même unité et les liens de l'application permettant de naviguer entre les entités.

Par la suite, plusieurs autres méthodes se sont inspirées de HDM. Ainsi, RMM (Relationship Management Method) [140] est devenue l'une des premières méthodes proposant un modèle et une démarche. RMDM (Relationship Management Data Model) est le modèle utilisé dans cette approche pour décrire les informations et les mécanismes de navigation des hypertextes. Ce modèle est basé sur le modèle EA et sur HDM. En parallèle, EORM (Enhanced Object Relationship Methodology) [30] préconise une approche orientée objet tout en s'inspirant du travail fait dans HDM sur l'hypertexte.

Suit OOHDM (Object-Oriented Hypermedia Design Method) [126], apparue en 1996. Ce métamodèle permet de décrire l'application à travers un modèle conceptuel décrivant les objets du domaine de l'application sous forme de diagramme de classe, un modèle de navigation décrivant le système hypertexte des structures d'accès ainsi que la navigation contextualisée entre ces structures. Les structures dans le modèle de navigation décrivent les données que l'utilisateur peut voir par rapport à un objet du modèle conceptuel. Le modèle conceptuel et le modèle de navigation semblent similaires à première vue. La différence entre les deux est que le modèle de navigation décrit les vues associées aux objets ainsi que les liens de navigation entre ces vues alors que le modèle conceptuel décrit la structure informationnelle (c'est-à-dire les attributs) des objets ainsi que les associations entre ces objets. La vue d'un objet peut être réduite à sa structure informationnelle d'où la similitude entre les deux modèles. OOHDM propose, aussi, le modèle de présentation abstraite décrivant d'une manière abstraite le style de l'application (agencement dans la page, couleurs, etc.) en utilisant des *templates* HTML.

La notion de processus, dans OOHDM, a été ajoutée plus tard (en 2004) [3] sous forme d'un enchaînement d'actions basiques décrivant les fonctionnalités offertes aux utilisateurs. Cet enchainement des actions décrit la navigation fonctionnelle de l'application. Dans le reste de ce document, ce type de processus sera identifié par le concept de processus d'implémentation.

Plusieurs autres méthodes orientées objet se sont succédées. Par exemple, WSDM (Web Site Design Method)[98] qui présente une approche centrée sur l'utilisateur, par opposition aux approches guidées par les données (HDM, RMM, OOHDM), et décrit l'application à travers un modèle décrivant les classes et les groupes des utilisateurs avec les activités qui leur sont permises. Un modèle conceptuel décrivant la partie statique de l'application, et enfin un modèle de navigation pour représenter les possibilités de navigation. HFPM (Hypermedia Flexible Process Modeling) [80] fournit les lignes directrices pour la planification et la gestion d'un projet Web couvrant la totalité du cycle de vie d'un tel projet. Il se compose de treize phases composées chacune d'un ensemble de tâches traitant d'une problématique spécifique. HFPM ne prescrit pas des techniques spécifiques, qui peuvent être choisies librement par les analystes et les développeurs. SOHDM (Scenario-based Object-oriented Hypermedia Design Methodology) [81], issu de la réflexion faite sur OOHDM, se distingue par son approche basée sur les scénarios pour établir les diagrammes d'activités permettant, *in fine*, d'obtenir le diagramme de navigation de OOHDM.

Au début des années 2000 OOWS (Object-Oriented Web-Solutions) [47] se base sur UML pour décrire les applications web. Comme les autres approches, OOWS propose le modèle conceptuel des objets de

l'application sous forme de diagramme de classes ainsi que le modèle de navigation centré sur les utilisateurs permettant de décrire à quoi l'utilisateur peut accéder. Ce modèle décrit la hiérarchie de l'application du point de vue navigationnel. Dans cette hiérarchie on trouve les systèmes qui sont composés de contextes de navigation. Un contexte de navigation décrit une vue sur les objets de l'application ainsi que les liens entre les différents niveaux de navigation. OOWS propose, aussi, le modèle de présentation permettant de décrire comment les éléments décrits dans les contextes de navigation seront présentés, par exemple le mode de pagination, l'ordre d'une liste ou la présentation d'un objet (tabulaire, registre, etc.).

Dans une approche unificatrice, UWE (UML based Web Engineering Methodology) [76, 116] se présente comme une solution plus aboutie et complète puisqu'elle se définit comme une extension conservatrice d'UML. UWE est définie comme un profil UML et s'inspire de RMM, OOHDM et WSDM par conséquent on retrouve le modèle conceptuel, le modèle de navigation et enfin le modèle de présentation [96]. Dans le modèle conceptuel, le concepteur décrit les objets du domaine sous forme de diagramme de classe. Dans le modèle de navigation, le concepteur décrit le système hypertexte en utilisant le concept de classe de navigation. Chaque classe de navigation est associée à une classe du modèle conceptuel. De même, chaque lien de navigation est associé à une association entre les classes du modèle conceptuel. Le modèle de navigation est aussi enrichi par les différents éléments d'accès, comme les menus ou les index, permettant de décrire la structure de la navigation (ou comment les nœuds sont visités). Enfin, dans le modèle de présentation le concepteur décrit comment les structures de navigation, décrite dans le modèle de navigation, sont présentées aux utilisateurs. Le modèle de présentation décrit l'interface utilisateur abstraite.

Le modèle de navigation de UWE a été étendu afin de supporter les processus en ajoutant un nouvel élément d'accès permettant d'activer un processus décrit par le diagramme d'activité UML. Ces processus décrivent l'enchaînement d'actions à un niveau technique non durable dans le temps. Ils sont réduits à l'exécution des fonctionnalités comme l'impression, le fait de s'identifier ou le fait de se déconnecter. Ces processus sont donc des processus d'implémentation.

Toujours en s'appuyant sur UML, OO- HMéthode[23] propose les mêmes modèles que les autres approches (conceptuel, navigation et présentation) et a permis à travers l'outil VisualWade d'offrir la possibilité de générer du PHP, du JSP ou de l'ASP ainsi que le modèle relationnel des données vers différents SGBD. Netsilon [150, 112], quant à lui, organise la modélisation des applications web autour de trois modèles ; le modèle métier, le modèle de navigation et le modèle de présentation. Le modèle métier permet de décrire, à travers le diagramme de classe UML, les objets du domaine avec leurs attributs

ainsi que leurs méthodes. Les méthodes de ces classes sont décrites à l'aide d'un langage appelé Xion proche du langage OCL et définies à un haut niveau d'abstraction. Le modèle de navigation permet de décrire une navigation dynamique. Cela en décrivant les pages web dynamiques. Dans ce modèle, une page web est composée de zones associées à des points de variation interactifs appelés centres de décision. Ces centres de décisions décrivent les actions à effectuer sur la page web selon le contexte d'utilisation, par exemple l'affichage d'un formulaire, l'affichage d'un fragment de page web, une requête à une base de données, etc. Enfin, le modèle de présentation qui est implicite et est laissé au choix des designers. Ainsi les designers conservent leurs outils favoris pour définir le style des pages. Dans ce cas, Netsilon se charge d'importer les différents fichiers du designer et propose un référentiel sur ces fichiers. À partir de ces trois modèles (métier, navigation et présentation), Netsilon permet une génération complète de l'application.

D'autres outils de génération de code pour des applications web prennent comme modèle d'entrée un modèle UML (avec ou sans Profil). Ces outils considèrent le métamodèle comme une notion implicite et dépendent souvent des technologies visées. Ce qui rend cette approche difficile à appréhender puisque UML n'est pas dédié à un métier donné. Nous citons par exemple le travail fait dans [117] ou [68](Web Application Extension pour UML). Le projet BluAge est un autre exemple [12] qui adopte une approche MDA pour générer le code des applications web à base de services depuis un modèle UML, profilé et structuré selon une structure spécifique. Dans ce projet, la partie visuelle est décrite par des templates en xHTML connectés au modèle métier.

W2000 [13] est une approche qui étend également la notation UML pour modéliser des éléments multimédias issus de HDM. L'analyse des besoins en W2000 est divisée en deux sous-activités : l'analyse des besoins fonctionnels et l'analyse des besoins de navigation. Cela est concrétisé par la modélisation des acteurs qui interagissent avec l'application, la modélisation des services et leur enchaînement pour chaque acteur et enfin la modélisation de la navigation. XIS (eXtreme Modeling Interactive Systems)[2] est un profil UML permettant de décrire une application interactive d'une manière générale. Ainsi, il est composé d'un modèle d'entités métier, d'un modèle de cas d'utilisation, décrivant les différentes opérations applicables sur les entités, avec la hiérarchie des acteurs, du modèle de navigation et enfin d'un modèle d'interaction. À partir de cette description, une génération partielle du code est faite et c'est au concepteur de le compléter afin de soutenir les besoins spécifiques. Dans le même sens NDT [92] se définit comme un profil UML décrivant le modèle conceptuel, le modèle de navigation et le modèle de l'interface utilisateur abstraite. Ces derniers modèles subissent des transformations de modèle afin

d'obtenir des modèles de base de l'application. Ces modèles sont ensuite modifiés pour ajouter des besoins plus spécifiques. Enfin, les modèles peuvent servir aux analyses et aux tests puisqu'ils se situent à un niveau technique détaillé.

Avec l'évolution de RMM lui permettant de s'adapter aux applications complexes, WebArchitect [136] a vu le jour. Il propose une extension de RMM pour décrire l'architecture des systèmes d'information basés sur le Web. WebML (Web Modeling Language) [1] est largement utilisé et représente une référence dans la modélisation des applications web complexes. Issu des travaux faits dans HDM et RMM, il propose quatre modèles pour décrire une application web : le modèle structurel EA décrivant les entités et les utilisateurs ; le modèle de composition décrivant les pages web de l'application et les informations qu'elles contiennent ; le modèle de navigation décrivant les liens entre les pages ; et enfin le modèle de présentation qui décrit l'aspect des pages. WebML a connu un succès dans le monde industriel grâce à l'outil WebRatio [21] qui permet, en plus de la modélisation, la vérification et la génération du code de l'application (par exemple J2EE, Struts) en s'appuyant sur des transformations de modèles. Les travaux faits sur WebML ont été la source d'inspiration du standard IFML [108](Interaction Flow Modeling Language) de l'OMG. IFML est centré sur la modélisation des interactions entre l'utilisateur et l'application. Le noyau fonctionnel (NF) est donc négligé.

Avec l'évolution technologique des approches génératives, des approches permettant la génération complète ou partielle du code ont vu le jour. Par exemple, [77] propose UWE4JSF une approche basée sur UWE pour générer, partiellement et semi automatiquement, des applications web. Son utilisation du langage OGNL rend cette modélisation dépendante de la technologie cible.

En se basant sur UWE ou OO-H pour la description fonctionnelle WebSA [122] rajoute une modélisation de l'architecture logicielle à base de composant. Un *merge* de ces modèles est ensuite généré pour enfin générer le code vers différentes cibles. D'autres travaux [70] ont porté sur une démarche purement à base de composants. Dans ce cas toute l'application est décrite par des composants sans une spécification de la logique métier de l'entreprise.

Dans [39], les auteurs décrivent une approche basée sur un modèle de scénarios. Ces derniers sont transformés en diagrammes de scénarios et diagrammes de cas d'utilisation. L'utilisateur doit modéliser les données et l'implémentation des méthodes au niveau du modèle, ce qui augmente la complexité du modèle. L'interface utilisateur, quant à elle, est décrite par Google Web Toolkit [141].

WebSpec [115], quant à lui, permet de décrire l'application à partir d'un modèle combinant les modalités des navigations et la structure des pages web. Ces modèles sont surtout utilisés à des fins de test et de simulation de l'interface utilisateur par exemple dans RUP [78]

ou WebTDD [114]. Pour la génération de code pour les CMS nous citons le travail de thèse fait dans [69] qui décrit comment générer des modules permettant d'adapter les CMS à partir d'une modélisation au niveau technique décrit par le métamodèle CMS-ML [24]. Ce métamodèle se présente comme une extension d'UML. Nous citons dans la même catégorie ArchGenXML [110], qui est une application permettant de générer des modules Plone [71] à partir d'un modèle décrivant le plan documentaire et le diagramme d'états représentant le workflow documentaire de chaque type de contenu.

Une des approches les plus répandues visant à modéliser les applications d'une manière générale est celle de la génération de code pour des applications web à partir des modèles décrivant les IHM. Dans le domaine des IHM, la plupart des approches de modélisation mettent en avant la partie visuelle et les interactions. Le NF y est décrit d'une manière implicite et adaptée aux besoins visuels. Dans cette approche, la description du NF est souvent basée sur un modèle de tâches hiérarchiques simples ne permettant pas de décrire des processus métier et des workflows complexes. D'autres remplacent le modèle de tâches par un modèle décrivant les processus métier associés à la dynamique et aux interactions de l'application comme, par exemple, dans [22]. Dans le domaine de la malléabilité des IHM, par exemple dans la thèse de [130], un travail bibliographique sur les outils de génération de code ou d'interprétation de modèle pour les IHM a été fait et abordé largement.

4.3.2 *Synthèse des approches adoptant une vision informatique*

Nous allons maintenant tenter de faire une synthèse des différentes approches, que nous venons de voir, adoptant une vision informatique.

Ces approches voient l'application web comme une architecture visuelle. Dans cette architecture, une application web est assimilée à un graphe de nœuds interconnectés. Les nœuds représentent les pages web et les arcs représentent les liens de navigation entre ces pages. Dans un graphe d'application web, nous pouvons identifier deux types d'arcs : les arcs de navigation structurelle permettant de passer d'une page à une autre et les arcs de navigation fonctionnelle permettant d'agir sur les données manipulées par l'application.

Afin de modéliser les applications web, les approches adoptant une vision informatique se basent sur cette architecture visuelle. C'est donc une modélisation centrée sur les aspects visuels d'une application web et non sur le besoin métier de cette application.

Comme vue dans la section précédente, la plupart de ces méthodes s'inspirent de la modélisation des hypertextes introduite par l'approche HDM. Cette approche répond au besoin de diffuser l'information avec les applications web de diffusion (voir section 4.2.1)

dans lesquelles le graphe ne contient que des arcs de navigation structurelle. Les sites web dans ce cas sont centrés sur le concept de page web (avec un contenu statique). La modélisation de ce type d'application reste raisonnable et assez simple.

Avec l'apparition des applications de type SIW (voir section 4.2.2) la navigation fonctionnelle est devenue de plus en plus présente et fortement contextualisée (c'est-à-dire que leur existence sur la page web dépend de plusieurs contraintes métier, par exemple le rôle de l'utilisateur connecté). Cela est dû au besoin d'automatisation des processus métier. Ainsi, la navigation fonctionnelle prend une place de plus en plus centrale dans ce type d'application.

En plus, une application de type SIW doit être adaptable par rapport aux utilisateurs et doit rependre aux besoins de chacun selon son rôle par rapport aux processus métier et selon ses préférences. La perception de l'application peut, donc, varier selon l'utilisateur connecté et selon les différentes contraintes associées aux données de l'application. Ceci rend une application de type SIW fortement dynamique.

Une modélisation d'une application de type SIW avec les approches existantes basées sur la modélisation des systèmes hypertextes peut augmenter la taille et la complexité des modèles produits dans certains cas et est impossible dans d'autres cas.

Dans cette modélisation, la navigation fonctionnelle est organisée par rapport à la structure des pages formant l'application et non selon l'enchainement logique des actions (c'est-à-dire les processus métier). Cet enchaînement logique des actions n'est, généralement, pas défini ou défini d'une manière implicite relevant plus des détails d'implémentation que des exigences métier, comme dans le cas de WebML. Cette description est faite sous forme de ce que nous appelons un processus d'implémentation. Les processus d'implémentation impliquent souvent le système et un seul utilisateur. Ces processus sont souvent instantanés couvrant une fonctionnalité comme un enchaînement de formulaires. De plus ces processus sont volatiles c'est-à-dire qu'ils sont perdus dès qu'ils sont interrompus. Ils sont, donc, assimilés à des fonctionnalités par exemple une fonctionnalité d'impression ou d'envoi de courrier électronique.

*Dans notre contexte, **implicite** signifie que la description doit être transformée afin d'obtenir le modèle concret du processus métier.*

Une telle description n'est pas pertinente pour les applications web collaboratives surtout celle de type SIW centré sur les processus métier. Une séparation entre les exigences métier et la présentation d'une application web est donc nécessaire.

4.3.3 *Une vision métier*

La volonté de remonter en abstraction a été formulée et la nécessité d'impliquer les acteurs métier dans la phase de conception a été prouvée et adoptée par plusieurs approches. Nous citons, par exemple, le travail fait par [142] qui rajoute la possibilité de modéliser des proces-

sus métier centrés sur les tâches, en utilisant la notation BPMN, pour générer un modèle OOWS situé à un niveau technique. Le modèle obtenu doit être complété par le designer ce qui permettra, *in fine*, de générer le code de l'interface utilisateur de l'application. Le modèle BPMN, quant à lui, est transformé vers une définition de processus exécutable WS-BPEL [43]. Dans cette approche, l'application générée permet à l'utilisateur d'exécuter une liste de tâches décrites dans le modèle des processus métier.

Le travail fait dans [134] propose une approche de prototypage rapide d'interface utilisateur exécutable. À partir d'un modèle de processus métier centré sur les tâches WBM [55] d'IBM, l'outil génère un modèle permettant de décrire l'interface utilisateur. Ce modèle est, ensuite, complété puis transformé en un modèle XML exécutable de l'interface utilisateur. Cette interface est enfin connectée à un noyau fonctionnel permettant une exécution des processus métier centrée sur les tâches. Le modèle métier d'entrée n'est pas obligatoire. Ainsi, le concepteur peut décrire directement son interface en utilisant le métamodèle intermédiaire. Ce dernier permet une description centrée sur l'utilisateur et ne se préoccupe pas des processus métier.

Sira et al dans [152] proposent une approche permettant la génération partielle des interfaces utilisateurs. Cette approche se base sur un modèle d'artefact, un modèle de processus métier centré sur les données (artefacts) et enfin les services. Les processus métier dans ce cas décrivent le cycle de vie de chaque artefact.

Dans le but de remonter en abstraction, WebRatio a proposé une méthodologie associant BPMN et WebML pour rendre l'application plus accessible aux acteurs métier. À partir d'un modèle BPMN, l'outil génère un modèle WebML encombré par les métadonnées décrivant implicitement les processus métier et leur exécution. L'application générée par WebRatio est équivalente à une interface utilisateur issue d'une solution BPM, par exemple Intalio [45]. Ainsi les processus métier sont simplement orchestrés. Parmi les solutions BPM distinguées par leur maturité sur le marché, nous citons BonitaSoft qui est un outil Open-source qui propose un éditeur de processus métier respectant le standard BPMN2.0 [129][6][107]. Cet outil propose un éditeur convivial de modélisation de processus métier BPMN, un moteur de workflow permettant d'exécuter les processus de l'entreprise et une application web proche d'une application mail pour l'interaction et le contrôle des processus. Ainsi les processus sont interprétés par l'outil et orchestrés par l'application web fournie. Dans ce processus de développement, l'utilisateur modélise ses processus métier, les connecte à son système d'information à travers des connecteurs puis exécute son application.

4.4 MODÉLISATION D'APPLICATION WEB AVEC WEBML

WebML (Web Modeling Language) [1] est un DSL dédié à la conception de sites Web complexes. Depuis son apparition, WebML n'a cessé d'évoluer pour s'adapter aux besoins récents et aux avancées technologiques. Au départ WebML se focalisait sur les site web de présence sur le Web et les sites web "catalogues" intensivement centrés sur les données. En conséquence, la modélisation permettait de décrire une application statique d'interface utilisateur permettant l'accès aux données. Toujours en se basant sur une description des systèmes hypertextes et sur une approche modulaire pour la spécification des interfaces utilisateurs, WebML se distinguait par l'introduction d'une modélisation à base de « composants », appelée « Unité ». Ceci a permis de rajouter des unités permettant d'agir sur les données (ajout, modification, suppression, etc.). Ainsi, on distingue les unités de contenu permettant de présenter l'information et les unités fonctionnelles permettant d'agir sur les données. Ces unités sont à la base des extensions apportées à WebML. En effet, chaque évolution a ajouté un ensemble d'unités et une légère modification du méta-modèle. De ce fait, WebML est passé d'une modélisation permettant la gestion des bases de données à une modélisation des applications riches en passant par une modélisation intégrant les processus métier. Sa présence dans le monde industriel à travers la plate-forme WebRatio lui a donné une visibilité et un succès au niveau industriel comme au niveau académique. WebRatio est un ensemble de plug-ins Eclipse. Il permet la spécification visuelle des modèles WebML ainsi que la génération de code J2EE à partir de ces modèles. Il supporte, aussi, le test de l'application, la génération des documentations, etc.

La figure 15 [20] décrit le processus de développement d'une application web avec WebML. Ce processus suit une démarche classique inspirée de différentes méthodes de développement d'application web. Ce processus, itératif et incrémental, permet de raffiner chaque étape le long de la conception. Dans ce processus la spécification des exigences métier n'est pas spécifique et est laissé aux choix des utilisateurs. Ces exigences sont ensuite transformées, manuellement, en un modèle WebML décrivant l'application à un niveau technique. À partir de ce modèle, l'application est implémentée afin de la tester ou de la déployer si les tests et les exigences métier sont validés.

Dans ce qui suit, nous détaillons les différents modèles WebML permettant de décrire une application web.

4.4.1 *Le modèle des données (structurel)*

Le modèle des données permet de décrire les entités gérées par l'application. Ce modèle se base sur la modélisation Entité/Association. Ainsi, chaque entité possède des attributs typés et des relations

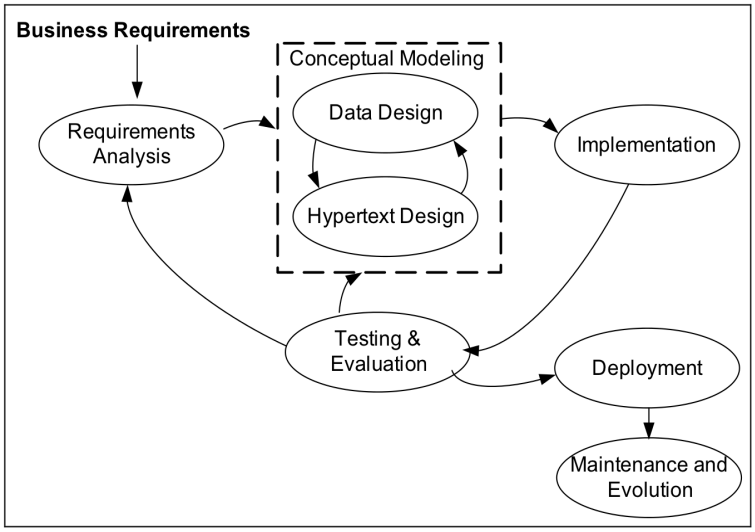


FIGURE 15: Le processus de développement d'une application web avec WebML [20]

bidirectionnelles avec les autres entités (sans attribut). Dans ce modèle, la notion d'héritage existe entre les entités bien que cette notion reste simple et ne peut être multiple. Dans WebML, une entité doit avoir un attribut « oid » unique. De ce fait, chaque instance d'entité est unique et identifiée par cet attribut. Dans le contexte de la modélisation des processus métier, des entités prédéfinies, comme « User » et « Group » pour la gestion des droits et « Process », « Activity », etc., pour la description de l'implémentation du moteur d'exécution des processus métier, sont spécifiés comme illustré sur la figure 16 [21].

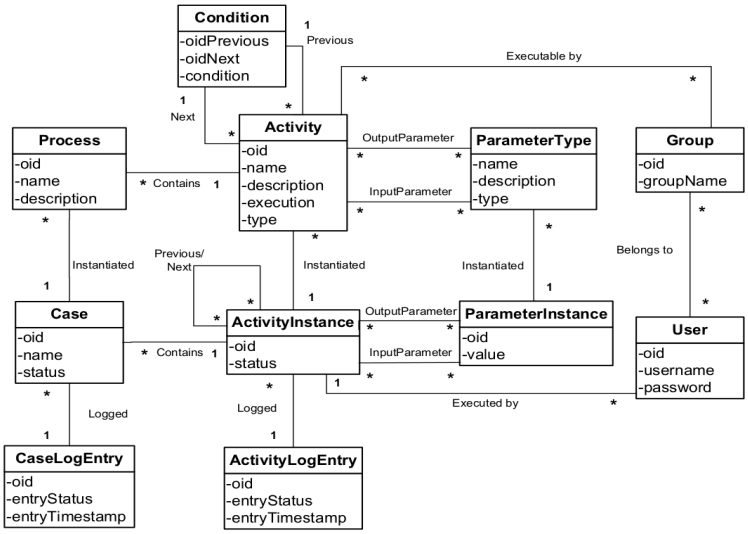


FIGURE 16: Le modèle des données WebML [21]

Le modèle des données peut être enrichi par des dérivations principalement utilisées pour importer des attributs entre entités reliées par une relation, définir des attributs calculés, etc. Afin de décrire ces

dérivations, WebML se base le langage OQL ou OCL. Par contre, aucune contrainte ne peut être posée sur les attributs ni les relations, à l'exception des contraintes structurelles concrétisées par les cardinalités sur les relations.

4.4.2 *Le modèle hypertexte*

Dans un modèle hypertexte, l'application web est vue comme un ensemble de pages web reliées entre elles par des liens hypertextes permettant la navigation entre ces pages. Ce modèle peut être décomposé en deux sous-modèles ; le modèle de composition et le modèle de navigation.

Le modèle de composition

Le modèle de composition permet de décrire la structure générale du site sous forme de pages web navigables et la structure de celle-ci en termes de contenu. Pour cela, WebML définit les concepts de page web, les unités de contenu « Content Unit » et les unités d'opérations « Operation Unit ». Ces unités offrent différentes possibilités, prédéfinies, pour intégrer du contenu dans les pages, par exemple la présentation détaillée d'une instance « Data Unit » ou la présentation simplifiée d'une liste d'instance « Index Unit ». Il existe, aussi, des unités pour la gestion du contenu, par exemple les unités de modification « Modify Unit » et de suppression « Delete Unit », etc. Afin de gérer les processus métier, des unités d'opération ont été rajoutées.

Le modèle de navigation

Une fois que le modèle des données et celui de composition sont définis, le modèle de navigation ajoute les liens permettant la connexion entre les pages et les unités afin de former une structure hypertexte. WebML décrit deux principaux types de liens :

- Les liens contextuels permettent de décrire l'enchaînement logique d'activation des unités, pouvant appartenir à des pages différentes. Cet enchainement logique permet de définir le sens de l'application finale. Un lien contextuel porte une information d'une unité source vers une unité destination. En parallèle avec la notion de composant, les informations présentent les paramètres d'entrée de l'unité destination. Ces informations dépendent du type de l'unité source : par exemple, pour une Data Unit, le contexte est l'OID de l'instance présentée par cette unité.
- Les liens non contextuels sont utilisés, essentiellement, pour passer librement d'une page à une autre indépendamment du contexte global de l'application. Ces liens permettent de déclencher une unité n'ayant pas besoin d'un contexte ou de passer directement à une page dans le site. Par exemple, un lien non

contextuel peut exister entre une page quelconque du site et la page d'accueil de celui-ci.

Ces liens peuvent être traduits en un bouton ou un lien hypertexte dans une application web. WebML permet, aussi, de décrire des liens non navigables et qui permettent, seulement, de porter de l'information entre les différentes unités.

4.4.3 *Le « modèle » de présentation*

La présentation permet de décrire l'apparence des pages du modèle de composition. WebML préconise l'utilisation des feuilles de style XSL (eXtensible Stylesheet Language) pour la spécification de la présentation. Le fichier XSL obtenu est ensuite attaché au modèle de contenu de l'application.

XSL est un langage du W3C (World Wide Web Consortium) permettant la description des feuilles de style. Ce langage est associé au langage XML.

4.4.4 *Conclusion sur WebML*

WebML est un DSL dédié à la spécification conceptuelle d'applications Web centré sur la modélisation des systèmes hypertextes. La notion d'unité est primordiale pour la description des exigences métier de l'application au niveau du contenu comme au niveau des opérations. Ces unités présentent un mécanisme d'extension dynamique pour WebML. Elles constituent un avantage certain au niveau de la réutilisabilité et de l'extensibilité [132]. Cependant, ces unités permettent une description, à base de composants, centrée généralement sur les détails techniques de l'implémentation et de la logique algorithmique de l'application web et non sur les exigences métier. Cela situe WebML à un niveau technique difficilement accessible par un acteur métier. Par exemple, dans le cadre de la modélisation des processus métier, WebML a proposé une extension qui porte sur l'ajout des entités et des unités permettant de décrire la logique métier d'une manière implicite. Cette extension décrit davantage le modèle d'exécution des processus métier au niveau technique et non les processus métier eux-mêmes [86].

Pour une évolution donnée, WebML cherche à rajouter un minimum d'unités et de concepts tout en décrivant la logique de leur utilisation. Selon nous, cette approche amène le concepteur à rajouter des détails redondants et difficiles à exprimer pour certains patterns de processus métier. Nous pensons que ces extensions présentent des détails techniques pouvant être déduits du modèle des exigences métier. Dans ce sens, WebRatio a proposé une démarche permettant de passer d'un modèle BPMN, situé à un niveau métier, à un modèle WebML plus technique, comme illustré dans la figure 17 [21]. Le modèle obtenu décrit une application BPM en introduisant des unités d'orchestration des processus métier.

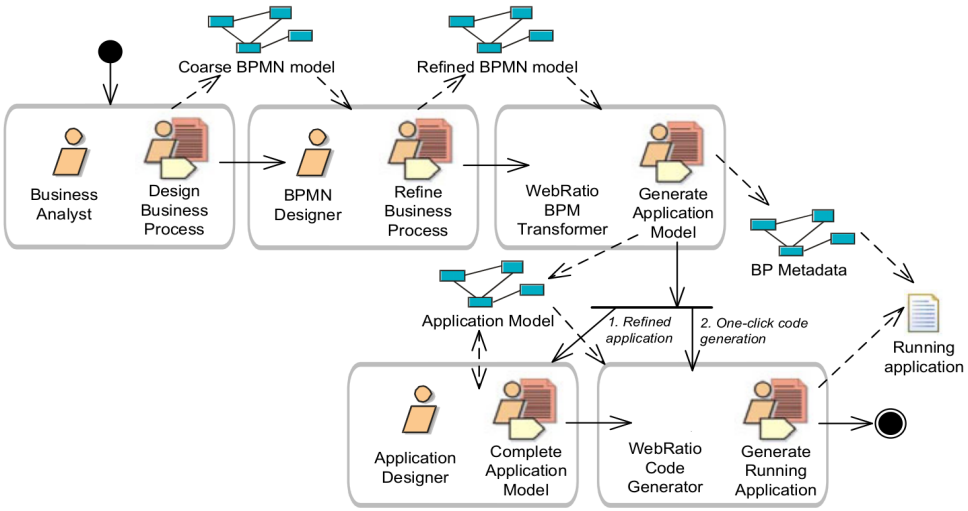


FIGURE 17: Processus de développement d'application web avec WebRatio [21] (notation en SPEM)

4.5 CONCLUSION

La modélisation des applications web n'est pas à son commencement et a pris comme point de départ la modélisation des systèmes hypertextes. Une modélisation qui a évolué pour couvrir les besoins des nouvelles applications web complexes. Avec l'aspect dynamique de plus en plus poussé des applications web, la notion de système hypertexte a changé d'une forme statique à une forme dynamique. Pour répondre à ce besoin, les modèles de composition et celui de navigation ont évolué et sont devenus de plus en plus contextualisés. Cette évolution a permis d'avoir des modèles, de l'application web, exprimant les besoins métier d'une manière implicite (c'est-à-dire noyés dans le modèle) et mis à plat (c'est-à-dire éclatés sur tous les modèles de l'application web).

Ces dernières années, avec les avancées technologiques dans le domaine de la génération de code et des transformations de modèles, plusieurs approches ont proposé des méthodologies visant la génération de code pour des applications web en commençant par une modélisation des exigences métier (processus métier ou scénarios). Dans ces cas, le modèle reste à un niveau secondaire et est destiné à des fins d'orchestration des processus métier. Dans la méthodologie suivie par les solutions BPM basées sur le web, l'application web est considérée comme une structure générique prédéfinie permettant le contrôle et l'orchestration des processus métier et ne prend pas la génération de l'application web comme une finalité.

Dans le chapitre suivant, nous montrons les limites de ces approches et nous proposons les critères permettant d'avoir un métamodèle ergonomiquement efficace.

5 LES LIMITES DES APPROCHES EXISTANTES

Sommaire

Dans le chapitre précédent 4, nous avons présenté une partie des approches de modélisation des applications web existantes. Nous avons classifié ces approches en deux catégories, qui sont : les approches de modélisation privilégiant une vision informatique, centrée sur la modélisation de la solution aux besoins, des applications web (AVI) et celles privilégiant une vision métier centrée sur la modélisation des besoins métier (AVM).

Le but de ce chapitre est de montrer les limites et les points faibles de ces approches et plus précisément des métamodèles proposés. Plusieurs approches d'évaluation de métamodèle existent. Dans la section 5.1 nous justifions notre choix qui est d'évaluer notre approche selon des critères bien définis. Puis dans la section 5.2 nous détaillons les critères que nous avons retenus. Enfin, dans la section 5.3 nous donnons les limites des approches existantes selon ces critères.

5.1 COMMENT ÉVALUER NOTRE APPROCHE ?

Nous avons identifié trois classes d'évaluation de métamodèles : l'évaluation en suivant des métriques ; l'évaluation en comparant les métamodèles et enfin l'évaluation par transformation entre les métamodèles à évaluer. Toutes ces approches d'évaluation de métamodèle présentent des points faibles, néanmoins la plus simple à mettre en place est celle permettant d'évaluer les métamodèles en les comparant. Pour évaluer notre approche, nous proposons de comparer notre métamodèle aux métamodèles des approches étudiées dans le chapitre 4. Pour cela, il nous faut trouver les critères pertinents permettant de comparer ces métamodèles. Dans cette section nous justifions notre choix concernant la comparaison des métamodèles. Pour cela, nous montrons les points faibles des approches d'évaluation que nous

avons identifiés. Puis, pour terminer, nous donnons le critère principal sur lequel nous nous basons pour évaluer notre métamodèle.

Un métamodèle possède deux relations directes et principales. La première est avec le concepteur (la personne chargée de décrire les modèles) et la deuxième avec le système à modéliser. Ni l'une ni l'autre ne forment un système formel. D'où l'impossibilité de calculer l'efficacité d'un métamodèle d'une manière générale. Malgré cela, nous trouvons dans la littérature des approches donnant des métriques pour des critères comme le critère de la complexité de construction d'un modèle conforme au metamodele à évaluer [131]. Cette métrique n'est pas, selon nous, suffisante et ne constitue pas une base pour l'évaluation globale d'un métamodèle. En effet, la construction d'un modèle peut être simplifiée par l'outil de modélisation.

D'autres études [26] [84] [5] [61] [48], proposent de comparer des métamodèles en se basant sur plusieurs critères. Le choix de ces critères est souvent fait par rapport au domaine, à l'application ou à la plate-forme cible. L'évaluation de ces critères reste relative par rapport aux richesses socioculturelles et les différences psychologiques, physiologiques et idéologiques (SPPI) [42] de l'homme ainsi que les avancées technologiques dans le temps. Ce qui rend ces critères spécifiques et instables.

Dans la littérature nous pouvons identifier un autre type d'approche d'évaluation des métamodèles se basant sur l'étude de perte d'information lors de la transformation d'un métamodèle vers un autre. Par exemple dans [85] les auteurs comparent, par transformations, le diagramme d'activité UML avec le métamodèle BPMN. Ce type d'approche permet de comparer l'expressivité des métamodèles. Selon nous, dans cette approche, deux métamodèles **MM1** et **MM2** sont équivalents si et seulement s'il existe au moins une transformation permettant de passer de **MM1** vers **MM2** sans perte d'informations et une autre permettant de passer de **MM2** vers **MM1** là aussi sans perte d'informations. Si on a une perte d'informations de **MM1** vers **MM2** alors **MM1** est plus expressif que **MM2**. Que dire dans le cas où on a une perte d'informations dans les deux sens ? Dans ce cas, une évolution permettant d'enrichir les deux métamodèles est nécessaire s'ils décrivent deux domaines différents ou un même domaine à des niveaux d'abstraction différents. Une unification et préférable dans le cas où les métamodèles décrivent, au même niveau d'abstraction, un même domaine. Selon nous, cette technique de comparaison est coûteuse en temps.

En l'absence d'une approche d'évaluation formelle et peu couteuse en temps, la comparaison par rapport à des critères que nous définissons reste notre seul choix. Comme vue précédemment cette approche d'évaluation de métamodèle à un point faible qui est la relativité de l'évaluation des critères choisit pour la comparaison des métamodèles. Selon nous cette relativité doit être minimisée. Cela en faisant le

bon choix des critères de comparaison. Maintenant, il nous faut trouver les bons critères de comparaison de telle sorte que l'évaluation de ces critères soit la moins relative possible.

La question de comparaison des produits d'une manière générale a été posée dans divers domaines. Par exemple dans [42] les auteurs discutent, largement, des théories à suivre et des critères à respecter pour avoir un produit logiciel convivial. Les critères discutés englobent le côté conceptuel (la présentation des connaissances) et le côté technique du produit (l'implémentation) par rapport à l'utilisateur. Ces critères peuvent être généralisés en un seul mot qui est l'ergonomie [19].

L'ergonomie est un concept qui peut être appliqué à tous produits (Applications web, outils de modélisation, voiture, etc.). Elle est un carrefour de différentes disciplines, comme les sciences humaines (sociologie, psychologie, etc.) ou les sciences de l'ingénieur, visant à rendre l'utilisation d'un produit, d'une manière générale, plus confortable, plus sécurisé et plus efficace pour les humains.

Dans ce qui suit, nous détaillons la notion d'ergonomie des métamodèle. Enfin, à partir de cette notion nous montrons les différents critères à respecter afin d'avoir un métamodèle ergonomiquement efficace.

5.2 QUELS SONT LES CRITÈRES D'UN MÉTAMODÈLE ERGONOMIQUEMENT EFFICACE ?

Nous allons, maintenant, justifier nos choix des critères à respecter afin d'avoir un métamodèle ergonomiquement efficace. Pour cela, nous nous basons sur la notion d'ergonomie. Selon nous, l'ergonomie est une entité divisible qui peut concerner plusieurs préoccupations d'un même produit. Cela parce qu'un produit peut être vu selon plusieurs préoccupations.

Dans cette section, nous allons commencer par donner les types d'ergonomies que nous avons identifiés. À partir de ces types d'ergonomies, nous allons pouvoir positionner nos critères. Puis nous allons détailler la notion d'ergonomie appliquée aux métamodèles selon le type d'ergonomie choisie. Cela va nous permettre de déduire nos critères permettant d'avoir un métamodèle ergonomiquement efficace.

Selon nous, une application (Application web, outils de modélisation, etc.) peut être découpée en deux parties complémentaires le **métier** et le **style** :

- Le **métier** se focalise sur les données et les actions du domaine métier (ou le noyau fonctionnel). Le but de cette partie est de garantir la présence des données ainsi que les actions au bon moment au bon endroit pour la bonne personne. L'utilisateur de l'application doit pouvoir, de ce fait, trouver les bonnes informations

(données ou actions) en respectant les différentes contraintes associées à son métier.

– Le **style** se focalise sur la présentation des données et l'interaction entre l'application et l'utilisateur. Le but de cette partie est de garantir une présentation ainsi qu'une interaction entre l'application et l'utilisateur les plus adaptées à la plate-forme ainsi qu'aux utilisateurs. La présentation et l'interaction dans ce cas décrivent comment les données sont présentées (position, couleur, police, etc.) et comment interagir avec ces données. L'interaction peut concerner un ensemble de données afin de déclencher un ensemble d'actions. Prenons comme exemple le déplacement d'un fichier dans un dossier en mode « glisser-déposer ». Dans ce cas, le fichier et le dossier sont les données, « Déplacer » le fichier dans le dossier est l'action, enfin le « glisser-déposer » est le type d'interaction.

La figure 18 illustre la relation entre l'IHM et notre découpage métier/style d'une application. Dans ce découpage, l'interface utilisateur (ou l'IHM) est constituée par des structures des données, dont les données sont issues de la partie métier, stylées selon la partie style.

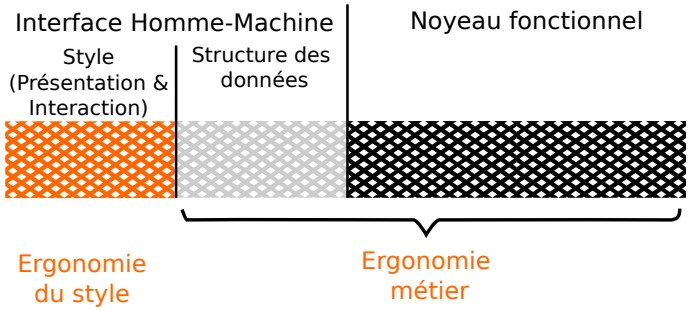


FIGURE 18: La relation entre le découpage métier/style et l'IHM des applications

D'un type d'application à un autre ou d'un type de plate-forme à un autre le métier ne change pas, par contre le style lui est variable. Nous parlons ici des applications multimodales. Une séparation entre le métier et le style est donc nécessaire. Si l'on transpose le découpage métier/style aux métamodèles, on peut dire que le métier correspond à la syntaxe abstraite ainsi que la sémantique. Le style correspond quant à lui à la syntaxe concrète et aux différents types d'interaction offerts par l'outil de modélisation.

La syntaxe abstraite d'un métamodèle est ce qui correspond aux instances des métaéléments (concepts et relations) du métamodèle. Cette syntaxe est unique.

La syntaxe concrète d'un métamodèle est une représentation graphique ou textuelle des concepts du métamodèle. Un même métamodèle peut avoir plusieurs syntaxes concrètes.

De cette distinction, nous pouvons conclure que l'ergonomie des applications est centrée sur deux points (voir figure 18) qui sont le métier (ergonomie métier) et le style (ergonomie du style). Dans notre cas, nous nous intéressons à l'ergonomie métier du métamodèle qui se focalise sur la syntaxe abstraite et la sémantique du métamodèle. Selon nous, l'ergonomie métier est un critère relativement stable dans le temps étant donné que le métier change peu en changeant de tech-

nologie. Afin d'évaluer notre métamodèle, nous nous basons sur la notion d'ergonomie que nous appliquons sur les métamodèles.

Le choix de l'ergonomie métier comme critère de base nous permet de réduire la relativité de l'évaluation de nos critères. Cela parce que l'ergonomie métier est peu sensible aux changements technologiques et aux changements de l'humain par rapport aux SPPI.

L'ergonomie peut être généralisée et est composée de critères qui sont l'**utilité** et l'**utilisabilité** [19] :

- L'**utilité** : Elle signifie ici « quelle est l'utilité du métamodèle ? » ou, autrement dit, « qu'est-ce que notre métamodèle permet de modéliser ? ». Dans notre cas, notre métamodèle permet de modéliser des portails collaboratifs (application web centrée sur les processus métier centrés sur les données). Ce critère nous permet d'identifier à quoi comparer notre métamodèle. Dans notre cas, la comparaison doit être faite avec les différents métamodèles permettant de décrire des applications web ainsi qu'avec les solutions BPM. L'utilité concerne la relation du métamodèle avec le système à modéliser.
- L'**utilisabilité** : Selon la norme ISO 9241 [144, 58], l'utilisabilité est vérifiée si l'utilisation du produit, d'une manière générale, est efficace et efficiente. Dans notre cas, l'efficacité signifie que le métamodèle doit permettre à l'utilisateur de modéliser ce qu'il veut modéliser avec aisance et facilité. Nous parlons ici de l'ergonomie métier. Nous nous intéressons donc à ce critère. L'efficience est le fait de modéliser rapidement avec un minimum d'erreur. Selon nous, cela tient plus de l'ergonomie du style et de l'ergonomie de l'interaction que de l'ergonomie métier. L'utilisabilité concerne la relation du métamodèle avec le concepteur.

L'efficacité d'un metamodele (que nous venons de voir dans le critère d'utilisabilité) peut elle-même être décomposée en deux critères principaux qui sont la **complétude** et l'**accessibilité** :

- La **complétude** du métamodèle : Elle signifie que le métamodèle doit permettre aux modèles de décrire tous les aspects du système. La complétude du métamodèle est favorisée par l'élargissement de l'espace d'expression (c'est-à-dire l'expressivité du métamodèle) et dépend fortement des connaissances récoltées dans un domaine donné. Cela conduit à la conclusion qu'un métamodèle doit être construit par/en collaboration avec les acteurs appartenant au domaine en question. Cela en faisant le bon choix entre la description explicite (c'est-à-dire que les concepts sont présents dans le métamodèle) ou implicite (c'est-à-dire que les concepts ne sont pas présents dans le métamodèle et peuvent être déduits automatiquement et sans ambiguïté de celui-ci) des concepts. La complétude du métamodèle ne doit pas être confondue avec la complétude du modèle. La complétude du modèle, elle, dépend de la sémantique définie au niveau du métamodèle.

L'incomplétude du modèle (un modèle qui n'est pas complet) ne présente pas une incohérence qui doit être corrigée, mais un manque qui peut être complété soit d'une manière automatique en utilisant des règles métier génériques ou/et d'une manière manuelle par intervention du concepteur. Pour compléter le modèle, il faut ajouter des informations suivant des conventions arbitraires standard ou non.

– L'**accessibilité** du métamodèle : Elle signifie que le métamodèle est facilement compréhensible par les concepteurs de modèles. Plus le métamodèle est accessible, plus il est adopté. En effet, la modélisation a pour but de simplifier le système réel et le rendre plus accessible aux concepteurs. Pour gérer la complexité du système, des contraintes doivent être respectées par exemple la séparation des préoccupations, la remontée d'abstraction ou l'adaptabilité de la syntaxe concrète à l'environnement SPPI des concepteurs. Selon nous, cette adaptabilité contribue à augmenter la notion d'accessibilité ce qui diminue la courbe d'apprentissage d'un métamodèle. Pour nous, l'adaptabilité tient plus de l'ergonomie du style et moins de l'ergonomie métier. Les autres critères (la séparation des préoccupations et la remontée d'abstraction) font partie de l'ergonomie métier.

Le non-respect du critère de complétude ou du critère d'accessibilité par un métamodèle implique la non-efficacité du métamodèle. Les critères de complétude et d'accessibilité ne tiennent pas compte de l'aspect technique d'implémentation du métamodèle, par exemple la portabilité du métamodèle sur plusieurs technologies.

D'une manière générale, pour qu'un métamodèle soit efficace, nous pensons qu'il doit être accessible et complet. L'efficacité d'un métamodèle est donnée par son niveau de complétude et son accessibilité. La mesure de complétude et d'accessibilité d'un métamodèle nécessite de le comparer à un métamodèle référence, un métamodèle dans lequel il y aurait tous les concepts décrivant le domaine à haut niveau d'abstraction avec une séparation des préoccupations parfaite. Naturellement, ce métamodèle référence n'existe pas. Il reste que nous pouvons faire des rapprochements et des comparaisons d'un métamodèle avec d'autres en suivant les critères de la complétude et de l'accessibilité donnés ci-dessus, même s'ils traitent de domaines différents. Les critères que nous avons choisis sont l'abstraction et la séparation des préoccupations pour l'accessibilité et enfin l'expressivité pour la complétude. Ces critères nous permettent de comparer l'efficacité des métamodèles. Selon nous, un haut niveau d'abstraction, une forte séparation des préoccupations ainsi qu'une expressivité élevée d'un métamodèle permettent d'avoir un métamodèle ergonomiquement efficace, du point de vue de l'ergonomie métier.

Dans ce qui suit, nous détaillerons ces critères et nous discuterons l'inadéquation des approches existantes (AVI et AVM) par rapport à ces critères.

5.3 L'INEFFICACITÉ DES APPROCHES EXISTANTES

Les approches de modélisation des applications web privilégiant une vision informatique (AVI) présentent un manque d'accessibilité. Quant aux approches privilégiant une vision métier (AVM), elles présentent un manque de complétude.

En effet, la finalité des approches AVI va du prototypage des interfaces utilisateurs à la génération partielle ou complète de l'application. Ces approches sont destinées, principalement, aux concepteurs informaticiens et se préoccupent davantage de la solution technique de l'application que du métier des utilisateurs. Aucune de ces approches ne part de la modélisation des entreprises centrée processus métier. En majorité, elles utilisent un métamodèle décrivant l'application sous forme de modèle de navigation entre les pages web rendant le modèle peu accessible aux acteurs métier comme dans le cas de OOWS ou WebSpec. D'autres utilisent des diagrammes tels que le diagramme d'activité afin de permettre la description des processus d'implémentation, comme dans le cas de OOHDM ou UWE. Ces processus se situent à un niveau technique décrivant l'implémentation d'une activité métier. Ainsi, nous avons constaté une fusion entre le processus métier et le processus d'implémentation. Cela a comme conséquence la difficulté de décrire des processus métier durables (s'étalant sur une longue période de temps) et complexes, ainsi que la difficulté de décrire les processus de contrôle et de gestion des processus. Dans le cas de WebML, la description des processus métier est implicite et est à un niveau technique d'implémentation. Les processus métier, dans ce cas, sont éclatés sur les différents modèles de navigation. En plus de cela, le concepteur doit gérer les concepts ajoutés automatiquement et décrivant la logique d'exécution des processus métier. Ces détails, selon nous, font partie de l'expertise technique qui doit être rajoutée automatiquement dans les modèles intermédiaires d'un processus de production de logiciels.

En ce qui concerne les AVM, ils ont introduit la modélisation explicite des processus métier, mais cela reste à un niveau secondaire par rapport à la modélisation de l'application. Dans ce cas, le passage par un modèle intermédiaire est obligatoire afin de compléter la modélisation de l'application web. Ce modèle intermédiaire constitue le niveau principal de la modélisation. Les solutions BPM quant à elles souffrent d'un manque d'expressivité. En effet, l'expressivité des solutions BPM est limitée à l'orchestration des processus métiers réalisés par des applications tierces ayant "un connecteur", c'est-à-dire une représentation dans le métamodèle et une interface logiciel connu par

le moteur d'exécution. Il n'est donc pas possible de couvrir des processus qui sont en dehors des domaines prévus par les concepteurs de ces outils et par conséquence de générer des applications métiers novatrices.

Dans la section précédente, nous avons identifié, pour améliorer l'efficacité des métamodèles, trois critères qui sont : la séparation des préoccupations, la remontée d'abstraction et l'élargissement de l'espace d'expression. Nous allons maintenant détailler ces critères.

5.3.1 *La séparation des préoccupations*

Les préoccupations sont les différents aspects d'un système. Leur séparation améliore la lisibilité et permet l'évolution des modèles en traitant chacune des préoccupations indépendamment les unes des autres.

Les questions permettant d'identifier les préoccupations d'un système sont : Qui, Quoi, Où, Quand, Comment, Combien, Pourquoi ? (QQOQCCP)[149]. Comme illustré sur la figure 19, ces questions sont au niveau métaméta par rapport à la classification des modèles de l'OMG vu dans le chapitre 3, on les appellera des métamétaquestions. À partir du métamétamodèle et les métamétaquestions, nous pouvons obtenir un métamodèle ainsi que des métaquestions associés à un domaine donné. Dans le domaine de la modélisation des entreprises, un exemple de métaquestion peut être « comment l'enchainement des activités est-il décri ? ». Le métamodèle, dans cet exemple, doit permettre de fournir les concepts permettant de répondre à cette question, par exemple en fournissant le concept « Processus-Métier ». Les métaquestions sont au niveau méta par rapport à la classification des modèles de l'OMG. Dans cette classification, le modèle répond aux mêmes questions que nous pouvons poser sur le système réel. Par exemple dans le domaine de gestions des factures, une question peut être « comment la facture est-elle traitée ? ». Le modèle, dans cet exemple, donne une réponse à cette question. Les questions sont donc au niveau modèle, par rapport à la classification des modèles de l'OMG.

Les métaquestions QQOQCCP peuvent être posées à plusieurs niveaux de détails. C'est pourquoi un sous ensemble des questions sur un niveau de détail peut former une question sur un niveau de détail supérieur.

La plupart des démarches existantes combinent les préoccupations. La forme la plus fréquente de cette combinaison est celle de la fusion entre les processus métier et l'architecture visuelle, qui se présente comme un modèle de navigation des écrans de l'application. Assurément, cette présentation exprime le métier de l'utilisateur d'une manière non structurée et implicite. En effet, cette présentation oblige à suivre l'enchainement des écrans pour comprendre la logique im-

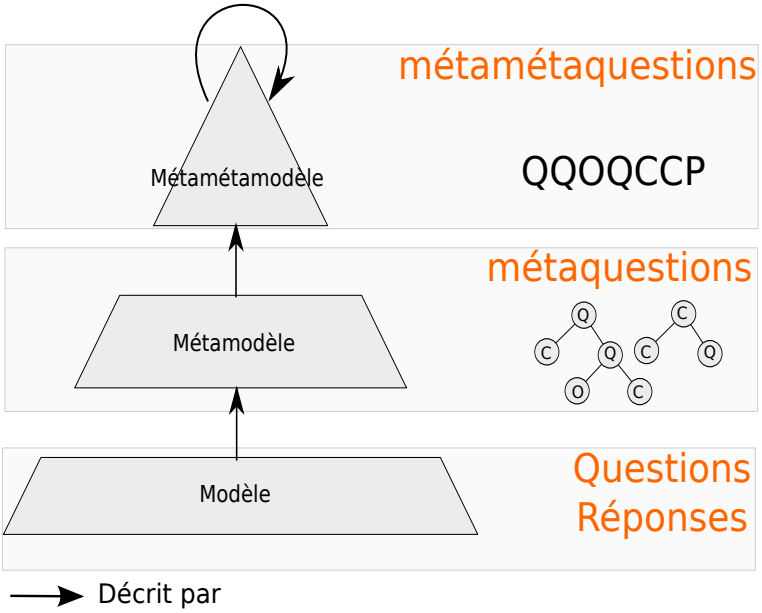


FIGURE 19: Classification des questions à poser sur un système selon la classification de l'OMG des modèles

plicite du métier qui n'est alors structuré que par la logique de l'application réalisée. Nous pensons que le modèle de navigation peut être déduit des modèles des exigences métier et de celui de l'aspect visuel. Nous pouvons comparer cette vision à celle de la marionnette à fils et du marionnettiste, où la marionnette représente l'aspect visuel de notre application et le marionnettiste représente le métier. Il suffit de regarder les mouvements du marionnettiste pour en déduire les mouvements de la marionnette.

D'autres formes de mélange des préoccupations existent comme celui entre la vue informationnelle et la vue fonctionnelle. En effet, dans la plupart des outils se basant sur des méthodes objet, par exemple UWE ou ArchGenXML, nous remarquons une fusion entre le « Quoi ? » et le « Comment ? » qui se présente comme une modélisation des objets de l'application tout en spécifiant les méthodes de chaque objet. Nous pensons que les méthodes des objets tiennent du « Comment ? » de l'application et peuvent présenter un détail technique qui ne doit pas être présent dans un modèle à haut niveau d'abstraction. Le modèle de la vue informationnelle doit décrire uniquement les objets de la collaboration et la relation structurelle entre ces objets (modèle entité/association). L'aspect comportemental des objets doit être décrit dans la vue fonctionnelle de l'application.

Un des mélanges des préoccupations le plus contraignants pour un acteur métier est celui des processus d'implémentation et des processus métier. Ce type de mélange est fortement présent dans UWE, à travers les processus de diagramme d'activité qui décrivent l'implémentation de l'activité (processus d'implémentation) avec les actions utilisateur (processus métier). Il est également présent dans WebML à travers

son diagramme de navigation permettant d'introduire les processus métier implicites éclatés sur les différents modèles de navigation et introduisant les détails de l'implémentation.

5.3.2 *La remontée d'abstraction et l'aération du métamodèle*

La remontée d'abstraction consiste à simplifier le métamodèle en enlevant les détails techniques et technologiques et en s'approchant le plus possible du langage humain, en utilisant les termes et les concepts du domaine à modéliser. Elle permet de rendre le modèle plus abordable et plus compréhensible par des non-informaticiens. En effet, les approches basées sur une vision informatique AVI présentent des modèles décrivant l'application à un niveau d'abstraction élevé par rapport aux langages de programmation, mais pas assez par rapport à l'utilisateur final (ici l'acteur métier). On peut comparer le résultat de ces approches au plan d'un bâtiment, qui reste difficilement compréhensible par le propriétaire et ne peut lui donner une image explicite de son futur bâtiment puisqu'il est fait par et pour des personnes du métier. Par contre, une maquette reste un moyen très efficace et plus accessible pour décrire les besoins du propriétaire.

La remontée d'abstraction est fortement associée à l'aération du métamodèle. En effet, dans un métamodèle une information peut être présentée d'une manière implicite ou explicite. Rendre explicites toutes les notions du système réel peut saturer le métamodèle, ce qui le rend difficile à maîtriser et rend les besoins métier difficiles à exprimer. Cela va à l'encontre d'un des principes fondamentaux de l'IDM qui est la simplification. Opter pour une expression implicite de certaines notions du système réel rend le métamodèle plus spécifique à un métier donné, plus lisible par le concepteur et donc plus accessible et plus aéré. Mais dans ce cas, les transformations de modèles jouent un rôle plus important dans l'analyse et la manipulation de ces modèles. Il faut donc mettre plus d'intelligence à ce niveau ce qui rend la maintenabilité des transformations plus difficile. En conséquence, un compromis est nécessaire entre l'aération du métamodèle et la maintenabilité des transformations de modèles.

La séparation des préoccupations est une solution à apporter à un métamodèle insuffisamment abstrait et/ou encombré. En effet, la remontée d'abstraction est favorisée par la séparation des préoccupations. Le passage d'un niveau d'abstraction à un autre converge vers la fusion des préoccupations et l'ajout des détails techniques et technologiques qui encombrent le métamodèle.

Les approches AVI, abordées dans la section 4.3.1, abordent une vue informatique des besoins métier en se basant sur les modèles centrés sur les systèmes hypertextes qui sont situés à un niveau technique. Le niveau d'abstraction dans ce cas n'est pas adapté aux acteurs métier.

D'autres approches, abordées dans la section 4.3.3, rajoutent la modélisation des processus métier (à un haut niveau d'abstraction) comme modèle de départ afin de générer un modèle technique (à un bas niveau d'abstraction) à compléter par le concepteur pour générer l'application finale. Il y a ici une remontée d'abstraction au niveau de l'approche, mais pas au niveau du métamodèle principal (ici le métamodèle technique) sur lequel la majorité du travail est fait.

5.3.3 *Élargissement de l'espace d'expression*

L'une des problématiques les plus connues dans le monde de la modélisation est l'expressivité. En effet, il est difficile de tout exprimer avec le modèle puisqu'il est destiné à simplifier le développement d'une application. Un concept, ou un pattern, dans un modèle, est considéré à un niveau d'abstraction et de détail très élevé par rapport aux langages bas niveau. On peut, donc, transformer un concept, ou un pattern, en plusieurs lignes de code. Nous pouvons comparer cette vision à celle de la grande brique et de la petite brique. On peut construire un bâtiment avec des formes et une finesse plus parfaites avec des petites briques qu'avec des grosses briques. La question est donc : comment approcher la finesse d'un langage à bas niveau avec le modèle ?

La réponse est de construire les grandes surfaces avec les grandes briques d'une manière la plus optimale et faire la finition avec les petites briques. Il faut trouver des techniques de modélisation et un langage abstrait pour augmenter le degré de finition et par conséquent élargir l'espace d'expression.

Les approches AVI ont en général une expressivité relativement élevée par rapport aux interfaces utilisateurs. En ce qui concerne le métier, ces approches montrent un manque d'expressivité pour des raisons différentes partant de l'absence des implémentations des actions ou méthodes des objets dans le cas de certaines approches orientées objet comme dans UWE jusqu'à l'impossibilité de décrire des workflows complexes impliquant plusieurs données, pertinentes, et à durée indéterminée comme dans OOHDM, OOWS etc. Les solutions BPM, appartenant aux approches AVM, quant à elles sont destinées à l'orchestration des processus métier à travers des applications prédéfinies. Le concepteur, dans ce cas, ne peut modéliser une application web spécifique aux besoins, par exemple un portail collaboratif.

5.4 CONCLUSION

Évaluer un métamodèle n'est pas une tâche facile étant donné ses relations directes avec le concepteur et le système à modéliser puisque ces derniers ne sont pas des systèmes formels. Cela ne permet pas

d'avoir un système de mesure de l'efficacité des métamodèles généralistes.

Pour évaluer notre approche, nous proposons de comparer notre métamodèle aux métamodèles utilisés dans les approches de modélisation des applications web que nous avons identifiées (AVI et AVM). Cette comparaison est basée sur des critères issus de la notion d'ergonomie que nous appliquons sur les métamodèles. Dans cette thèse, nous nous intéressons à l'ergonomie métier qui décrit la présence de l'information (donnée ou action métier) au bon moment, au bon endroit, pour un utilisateur selon les différentes contraintes métier. Pour un métamodèle, l'ergonomie métier se limite à la sémantique et à la syntaxe abstraite. L'ergonomie métier est centrée sur le critère de l'efficacité. Le choix de l'ergonomie métier comme critère de base nous permet de réduire la relativité de l'évaluation de nos critères. Cela parce que l'ergonomie métier est peu sensible aux changements technologiques et aux changements de l'humain par rapport aux SPPI.

Pour évaluer l'efficacité d'un métamodèle, nous avons choisi différents sous-critères que doit vérifier ce métamodele. Ces sous- critères sont : la complétude qui est favorisée par l'expressivité du métamodèle et l'accessibilité qui est favorisée par la séparation des préoccupations ainsi que la remontée d'abstraction.

Les AVI souffrent d'un manque d'accessibilité, car malgré la tentative de certaines approches comme WebML d'introduire la modélisation des processus métier, ces processus restent inaccessibles par les acteurs métier et sont difficilement exprimables. Ce manque d'accessibilité rend ces approches peu efficaces du point de vue des acteurs métiers. Les AVM quant à eux sont caractérisées par leur manque d'expressivité qui rend ces approches aussi peu efficaces que les AVI par rapport aux acteurs métiers.

Il est donc nécessaire de créer un équilibre entre l'accessibilité et l'expressivité afin d'augmenter l'efficacité. Par conséquent, nous devons augmenter l'expressivité de notre métamodèle tout en restant à un haut niveau d'abstraction. Dans le chapitre suivant, nous décrivons notre approche basée sur le métamodèle MACoP permettant de décrire les portails collaboratifs à un haut niveau d'abstraction. Dans MACoP nous nous sommes intéressés à l'ergonomie métier des portails collaboratifs. L'ergonomie du style ainsi que celle de l'interaction n'est pas traitée dans cette thèse et sera le sujet de futurs travaux. Cette séparation est utile dans un contexte où le métier ne change pas en changeant de plate-forme par exemple. Par contre, le style et les types d'interaction peuvent changer. De ce fait, cette séparation nous permettra de faire évoluer les portails collaboratifs selon le type de plate-forme visé en changeant simplement le modèle de style et celui d'interaction sans modifier le modèle métier.

Troisième partie

CONCEPTION DES PORTAILS COLLABORATIFS

6

MODÉLISATION DES PORTAILS COLLABORATIFS

Sommaire

Afin de concevoir nos portails collaboratifs, nous nous basons sur une approche IDM. Dans une telle approche, le modèle joue un rôle central. Nous proposons un DSL (Domain Specific Language) que nous appelons MACoP (Modeling and Analysis of Collaborative Portal). MACoP permet une modélisation centrée sur les processus métier des portails collaboratifs. Cette modélisation est à un haut niveau d'abstraction tout en ayant une expressivité élevée.

Dans ce chapitre, nous commençons par un exemple simple de notre approche, expliquée dans la section 6.1, puis nous décrivons

notre approche d'une manière générale dans la section 6.2. Ensuite, nous détaillons le métamodèle MACoP avec ses différentes vues dans les sections 6.3 et 6.4. Après nous, discutons des différentes extensions apportées à MACoP pour supporter la génération complète du code pour les portails collaboratifs dans la section 6.5. Enfin, dans la section 6.6, nous parlons de la réutilisation de modèle dans MACoP.

6.1 VUE D'ENSEMBLE DE NOTRE APPROCHE SUR UN EXEMPLE SIMPLE

Avant d'entrer dans les détails de notre proposition et de ses justifications, nous allons expliquer succinctement notre approche sur un exemple simple. L'exemple représente une collaboration entre un client et un vendeur. La collaboration se limite à un processus métier de facturation : le vendeur crée une facture en spécifiant son titre ainsi que le montant total, ensuite, le client choisit entre payer ou refuser la facture.

La modélisation de cette application nécessite la *modélisation des objets de collaborations*, la *modélisation des rôles* et la *modélisation des processus métiers*. À partir de ces modèles, l'application est *générée* et peut être immédiatement *utilisée* par les utilisateurs.

6.1.1 *Modélisation des objets de collaborations*

Les *objets de collaborations* sont les objets au sens large participants à la collaboration. On trouve ainsi parmi les objets de collaboration les objets contenant de l'information (comme des documents ou des factures), les utilisateurs participants à la collaboration ainsi que l'application servant de support à cette collaboration.

Dans notre exemple, nous identifions comme objets de collaboration : l'application, les utilisateurs de l'application (client et vendeur) et les factures. Une facture a un titre permettant de l'identifier et un montant total. L'objet de collaboration *utilisateur* permet de stocker, au sein de l'application, des données relatives à l'utilisateur, comme son nom, son adresse, voir son mot de passe.

La figure 20 montre la modélisation de nos objets de collaboration dans notre diagramme d'objets de collaboration, inspiré du diagramme de classes UML. Chaque classe représente un type d'objet de collaboration. Les attributs d'une classe représentent les données associées à l'objet de collaboration. Ainsi, nous associons un titre et un montant à la classe Facture. Nous distinguons, ici a l'aide de stéréotypes, la classe représentant l'application (stéréotype « application »), les classes représentant de l'information (stéréotype « information ») et les classes représentant des utilisateurs humains (stéréotype « human »). Ceci servira lors à la phase de génération. Les objets de collaborations doivent tous être accessibles, directement ou indirectement,

à partir de l'objet représentant l'application. C'est pourquoi la classe **MonApplication** permet de naviguer, à travers ces deux associations, vers la classe **Facture** et la classe **Utilisateur**. Les instances des objets de collaborations forment un graphe d'objets dont le point d'entrée est l'objet stéréotypé « application », ici **MonApplication**. Nous appelons ce point d'entrée la racine de l'application.

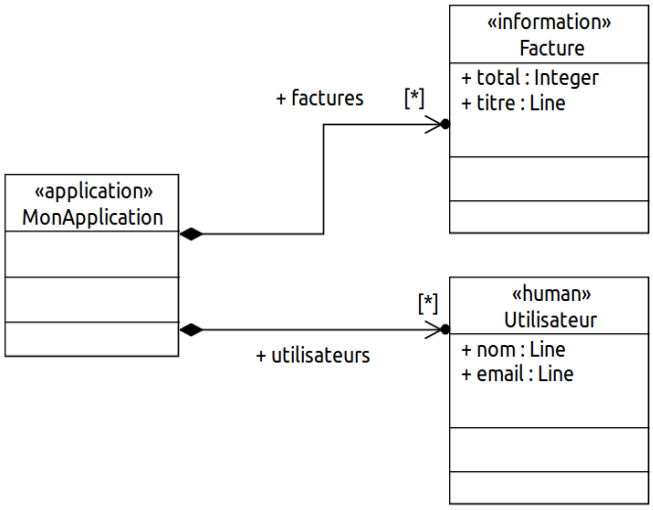


FIGURE 20: Les Objets de collaboration de l'exemple de facturation

6.1.2 *Modélisation des rôles*

Un rôle permet de définir une catégorie d'utilisateurs ayant des caractéristiques communes. Parmi ces caractéristiques, on trouve les processus métier utilisables par le rôle, les actions potentiellement déclenchables, les instances des objets de collaboration sur lesquels l'utilisateur a le droit d'agir.

Dans notre exemple, tous les vendeurs partagent les mêmes caractéristiques : entre autres, ils ont le droit de créer et de modifier des factures. De même, tous les clients peuvent choisir entre les actions de payer la facture ou de la refuser. Nous avons donc les rôles *Vendeur* et *Client*.

La figure 21 montre la modélisation de nos rôles dans notre diagramme de ressources, inspiré du diagramme des cas d'utilisation d'UML. Nous retrouvons nos deux rôles Vendeur et Client. Nous avons ajouté un rôle abstrait commun, *Collaborateur*, dont héritent les deux autres rôles. Ce rôle commun permet de regrouper les caractéristiques communes aux vendeurs et aux clients.

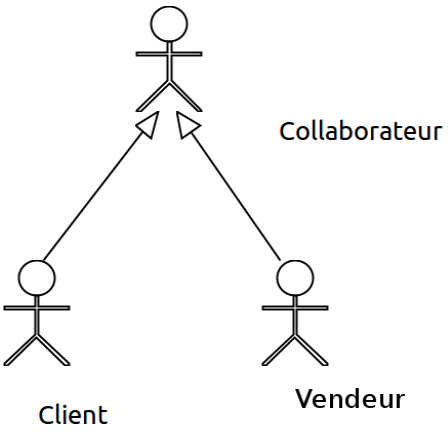


FIGURE 21: Les rôles de l'exemple de facturation

6.1.3 *Modélisation des processus métier*

La description d'un processus métier est constituée de l'ensemble des actions constituant le processus. Le processus métier est modélisé sous la forme d'un graphe d'actions. Une action représente un ensemble d'opérations élémentaires agissant sur un ensemble d'instances d'objets de collaborations.

Chaque action est associée à un rôle spécifiant quelle catégorie d'utilisateurs peut la déclencher. Une action est aussi associée à des contextes permettant de spécifier un certain nombre de propriétés comme : l'instance sur laquelle l'action peut s'appliquer ; les conditions requises pour que l'action puisse être déclenchée ; les instances des objets de collaborations qui seront modifiés par l'action ; les instances des objets de collaboration qui seront retournées par l'action.

Un processus métier est aussi associé à un concept, appelé *datastore*, permettant de stocker les instances des objets de collaboration manipulées. Dans ce *datastore*, les instances sont identifiées par un nom. Les actions du processus métier peuvent ainsi manipuler les instances du datastore, en créer de nouveaux ou en détruire.

Dans notre exemple, nous avons à priori un seul processus métier qui est le *processus de facturation* décrit plus haut. Cependant, nous devons ajouter au moins deux autres processus métiers : un premier afin de pouvoir *visualiser l'ensemble des factures* ; et un autre permettant de *gérer les utilisateurs*.

Processus de facturation

Ce processus permet à un vendeur de créer une facture, puis à un client d'accepter la facture, ou de la refuser.

La figure 22 montre la modélisation de ce processus dans un diagramme de type BPMN. Le diagramme comprend deux parties horizontales (lanes) associées chacune à un rôle. La partie haute est as-

sociée au rôle *Vendeur* : elle contient les actions déclenchable par le vendeur, ici *Créer une facture*. La partie basse est associée au rôle *Client* : elle contient les actions déclenchables par le client : *Payer*, ou *Refuser*. Le diagramme indique que le processus commence par l'action *Creer une facture*, déclenchable par le vendeur. Ensuite, une fois l'action finie, le client peut déclencher au choix l'action *Refuser* ou *Payer*. Après qu'une de ces deux actions soit effectuée, le processus se termine.

L'action *Créer une facture* permet de créer une facture et de saisir son titre et son total. L'action ne peut être déclenchée par l'utilisateur que si l'instance d'objet de collaboration qu'il étudie est l'instance nommée **monApplication**. Ceci est spécifié par le filtre **monApplication** (stéreotype « instanceSpecificationFilter »), dans lequel l'expression indique l'instance requise (désignée par son nom). Le second filtre, **facture**, indique que l'action a besoin d'une seconde instance pour être déclenchable ; et que cette instance doit avoir ses attributs **titre** et **total** renseignés (c'est à dire non nul). Le filtre indique aussi que l'instance de l'objet doit être dans l'état **néant**, c'est-à-dire qu'elle ne doit pas exister ! Dans ce cas, le système interprète le filtre comme une demande de création de l'instance. Une nouvelle instance de l'objet spécifié sera créée lors du déclenchement de l'action, et le système proposera à l'utilisateur de remplir les valeurs des propriétés spécifiées par « attributes ». L'instance créée sera stockée dans le *datastore* sous le nom *facture*, grâce à l'expression.

Les actions *Payer* et *Refuser* sont elles aussi associées à un contexte de réalisation (un seul est montré dans le diagramme) spécifiant que ces actions sont déclenchables si une instance nommée *facture* est présente dans le *datastore* du processus. Ces actions sont donc potentiellement déclenchables uniquement après la création de la *facture*, et donc après l'appel de l'action *créer facture*.

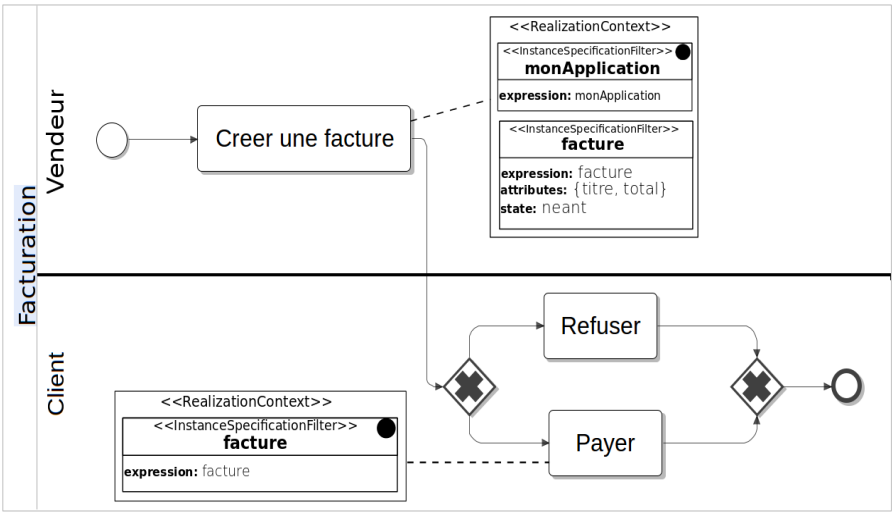


FIGURE 22: Le processus de facturation

Processus visualiser les factures

Ce processus permet à un vendeur ou un client de visualiser l'ensemble des factures existantes dans l'application. Ce processus est nécessaire, car dans notre approche, la navigation structurelle, c'est-à-dire la navigation permettant de naviguer entre les pages du portail collaboratif, est considérée comme une action métier (voir 6.2.2).

La figure 23 montre la modélisation de ce processus. Ce processus est commun aux vendeurs et aux clients : nous associons alors les actions au rôle commun *Collaborateur*. Le processus contient une unique action *Voir l'application* dite **automatique**, car le système la déclenche automatiquement lors de l'exécution du processus. L'icône représentant des engrenages permet de différencier, dans les diagrammes, les actions automatiques des actions déclenchées par l'utilisateur. Cette action est associée à un contexte d'étude (stéréotype « PostStudyContext ») permettant de décrire les données (stéréotype « Data ») qui doivent être renvoyées, par l'action, à l'utilisateur. Dans notre exemple nous spécifions que l'action renvoie la liste des factures, ce qui est exprimé par l'expression OCL (« **monApplication.factures** »). Le système affichera la liste des instances retournées par l'évaluation de l'expression.

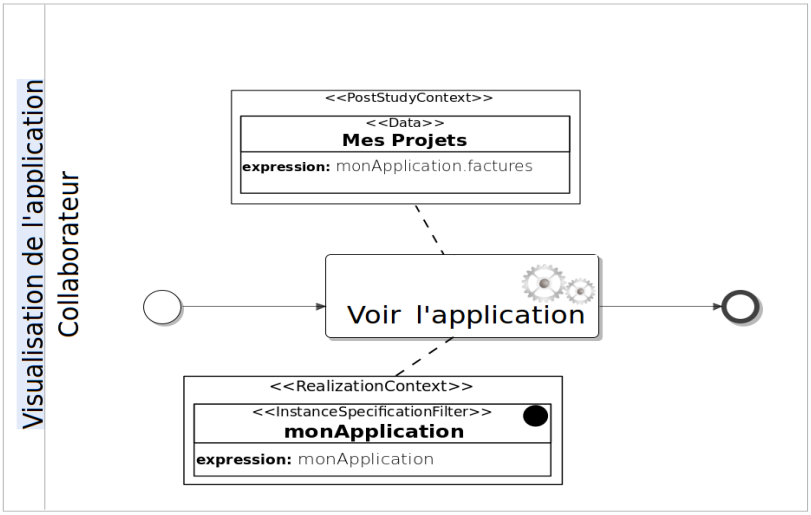


FIGURE 23: Le processus de visualisation de l'application

Le processus que nous venons de décrire permet de visualiser la liste des factures. Sur le même principe, nous avons aussi modélisé un processus pour visualiser une seule facture (non décrit dans cette section). Dans ce processus, un **Collaborateur** a le droit de visualiser les attributs d'une facture (**titre** et **total**).

Processus de gestion des utilisateurs

Ce processus permet de gérer les utilisateurs pouvant utiliser l'application, c'est-à-dire d'ajouter, de modifier ou de supprimer des utilisateurs. Ce processus peut être décrit finement dans un modèle. Mais dans le cadre de notre exemple, nous avons choisi d'utiliser le processus par défaut. Pour cela, il suffit de ne pas décrire le processus, et de laisser le mécanisme de génération de l'application ajouter automatiquement un processus de gestion des utilisateurs. Ce processus par défaut fournit la gestion de base, comme l'inscription de nouveaux utilisateurs, l'attribution des rôles ou la modification de ses données par un utilisateur. Ce processus de gestion des utilisateurs s'appuie sur les objets de collaboration de type « human ».

Ainsi, dans notre exemple, le processus de gestion des utilisateurs par défaut s'appuiera sur l'objet de collaboration **Utilisateur**. Il permettra, lors de son exécution, de créer des **Utilisateurs** et de renseigner les attributs associés (**nom** et **email**), mais aussi de spécifier les rôles que l'on veut associer à cet utilisateur. Le processus fournit aussi des actions déclenchables par l'utilisateur, lui permettant de modifier ses données, que nous appelons le *profil* de l'utilisateur.

6.1.4 *Génération de l'application*

Notre exemple est maintenant complet et nous pouvons générer le code de l'application. Pour cela, il suffit de lancer le script de génération. Aucune intervention n'est nécessaire dans le code généré. Le générateur de code analyse les modèles afin d'en extraire entre autres les objets métiers, la logique d'enchainement des actions, les conditions d'activations des actions et les données à présenter à l'utilisateur. Le générateur de code complète les modèles s'il détecte des "situations par défaut".

Le générateur de code ne génère pas de pages web, mais du code permettant de générer dynamiquement les pages web lors de l'exécution de l'application en fonction de l'instance d'objet de collaboration étudiée.

L'application est maintenant opérationnelle. L'utilisateur peut la déployer sur un serveur web, et accéder à la page principale (d'index).

6.1.5 *Le portail collaboratif en action*

Nous allons maintenant décrire comment fonctionne le portail collaboratif généré.

Notre approche du portail collaboratif comprend d'un coté un graphe d'instances des objets de collaborations, décrit par la modélisation des objets de collaborations, et de l'autre coté un ensemble de processus métiers contenant des actions applicables sur les instances

des objets de collaboration. Ces actions ont en outre des conditions d'activations.

L'utilisateur navigue dans le graphe des instances des objets de collaboration afin d'en étudier les instances qui l'intéressent. Quand l'utilisateur sélectionne une instance à étudier, le système détermine toutes les actions applicables à cette instance. Parmi ces actions, on distingue les actions déclenchables par l'utilisateur et les actions à déclenchement automatique. Les premières sont présentées à l'utilisateur dans un tableau de bord, ce qui lui permettra de les déclencher. Les secondes sont déclenchées immédiatement par le système, et leurs résultats sont concaténés afin de former "le contenu" qui est présenté à l'utilisateur. C'est ce contenu qui permet de montrer de l'information à l'utilisateur.

Chaque fois que l'utilisateur étudie une instance d'objet de collaboration, le système calcule la page à afficher, qui est donc formée d'un tableau de bord contenant les actions déclenchables, d'un explorateur indiquant l'instance d'objet de collaboration que l'on étudie, et d'un contenu présentant des informations.

Quand l'utilisateur choisit de déclencher une action, le système détermine alors les données requises par cette action. Il les affiche dans un formulaire, ce qui permet à l'utilisateur de les renseigner. L'utilisateur remplit le formulaire, et le valide. Ceci exécute l'action proprement dite. Une fois l'action exécutée, le système recalcule la page à afficher (tableau de bord, explorateur et contenu), en incluant dans le contenu les résultats spécifiés par l'action. Ainsi, l'utilisateur peut prendre connaissance du résultat de l'action, et choisir parmi les actions suivantes.

Comme on le voit, c'est le choix de l'instance d'objet de collaboration étudiée qui détermine le contenu de la page présentée à l'utilisateur. La page propose les actions possibles à partir de cette instance d'objet de collaboration. L'utilisateur peut changer l'instance étudiée grâce à l'explorateur, et aussi en cliquant sur des liens générés dans la partie "contenu" de la page. De plus, l'exécution d'une action, peut elle aussi, changer l'instance d'objet de collaboration étudié.

6.1.6 *Scénario d'utilisation de l'application générée*

Dans le portail collaboratif généré, chaque page est composée de trois parties principales : le tableau de bord permettant de montrer l'ensemble des actions que l'utilisateur peut effectuer ; le navigateur permettant de montrer le chemin allant de la racine de l'application jusqu'à l'instance d'objet de collaboration qui est étudiée et enfin, le contenu de la page permettant d'afficher les données associées à l'instance d'objet de collaboration qui est étudiée, par exemple les formulaires ou les valeurs des paramètres de l'objet de collaboration.

Une fois l'application déployée, l'utilisateur invoque la page principale. À partir de cette page, il est possible de s'authentifier, ou d'accéder au processus de gestion des utilisateurs. Nous allons maintenant décrire l'utilisation de l'application sur un scénario simple.

Un utilisateur commence par s'authentifier sur l'application. L'instance étudiée est, alors la racine de l'application. L'utilisateur est considéré comme anonyme ce qui lui donne le droit de s'inscrire. La figure 24 montre la racine de l'application **monApplication** avec l'action **Inscription Utilisateur** du processus de gestion des utilisateurs dans le tableau de bord.

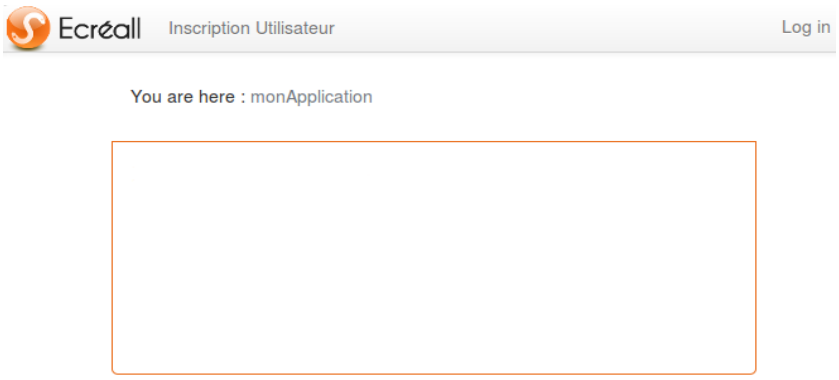


FIGURE 24: Page d'accueil pour les anonymes

L'utilisateur choisit de s'inscrire en cliquant sur l'action **Inscription Utilisateur**. Le système lui montre le formulaire afin d'introduire les données d'inscription, par exemple le nom l'email, etc. Ainsi un utilisateur peut s'inscrire en tant que **Bob** et un autre en tant qu'**Alice**. La figure 25 montre la page associée à l'action **Inscription Utilisateur** dans laquelle un utilisateur anonyme doit saisir son **nom** et son **email** afin de créer une instance du type **Utilisateur**. Dans notre cas, nous avons l'inscription de l'utilisateur **Alice**.

Le processus de gestion des utilisateurs inclue le rôle Administrateur permettant d'administrer les membres. Un utilisateur ayant le rôle Adminstrateur trouve les profils créés en étudiant la racine de l'application et leur attribue des rôles. La figure 26 montre la page associée à l'action **Assigner des rôles** dans laquelle l'administrateur doit choisir la liste des rôles à assigner à l'utilisateur. Dans notre cas, l'administrateur assigne le rôle **Client** à **Alice**. De même, il assigne le rôle **Vendeur** à l'utilisateur **Bob**.

Bob peut désormais se connecter en tant que **Vendeur** en cliquant sur l'action login du tableau de bord. La figure 27 montre la page permettant de se loger au portail collaboratif.

Une fois **Bob** connecté, le tableau de bord ainsi que le contenu de la page associée à la racine de l'application change. Le tableau de bord affiche l'action **Creer une facture** lui permettant de créer une

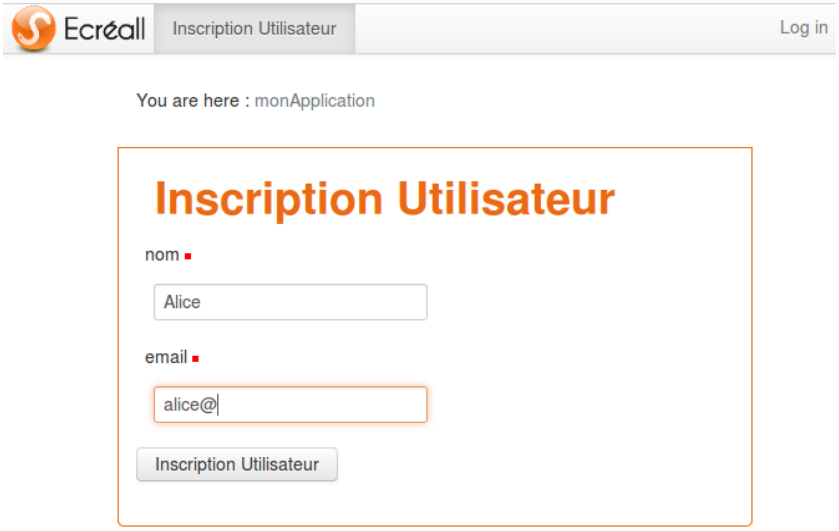


FIGURE 25: L'inscription de l'utilisateur **Alice**

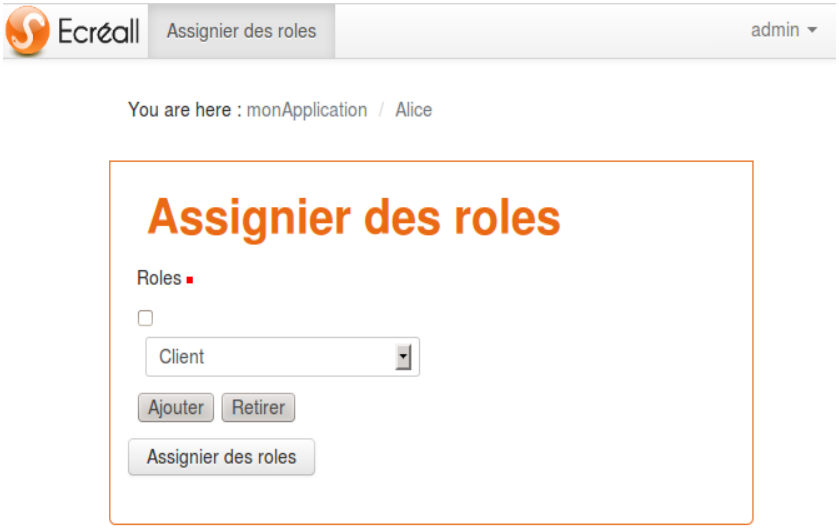


FIGURE 26: L'action permettant d'assigner des rôles au *profil* d'**Alice**

facture dans la racine de l'application. La figure 28 montre le formulaire généré pour cette action. **Bob** rempli le formulaire et valide.

Une nouvelle instance de **Facture** est créée dans la racine de l'application et devient accessible par les collaborateurs. La figure 29 montre la liste des factures de l'application vue par **Bob**. Cette liste correspond à ce que l'action **Voir l'application**, du processus de visualisation, montre comme données.

Ainsi, il est possible à **Bob** d'accéder à la facture créée. La figure 30 montre le détail de la facture vue par **Bob**. La page montre un tableau de bord vide pour **Bob**, car, comme décrit dans le modèle de facturation, le vendeur ne peut que créer une facture.

En parallèle, l'utilisateur **Alice** se connecte et trouve sur la racine de l'application la facture créée par **Bob**. **Alice** clique dessus afin d'y ac-

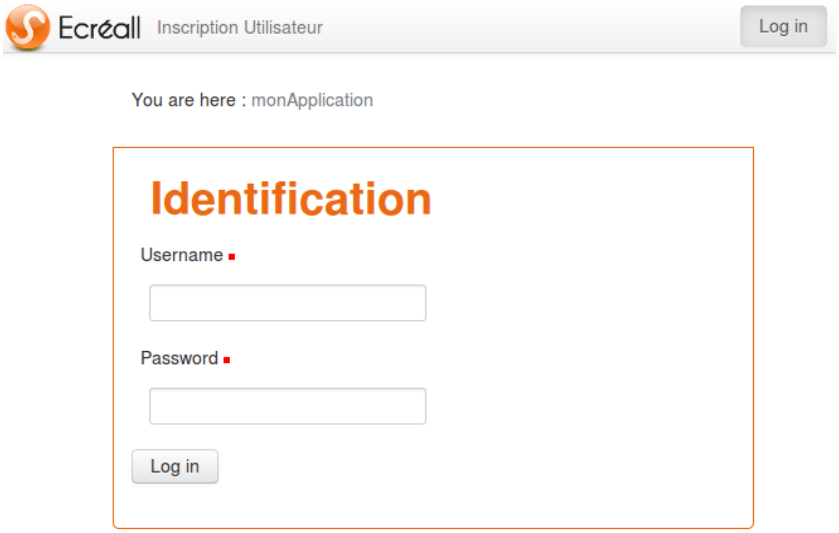


FIGURE 27: La racine de l'application

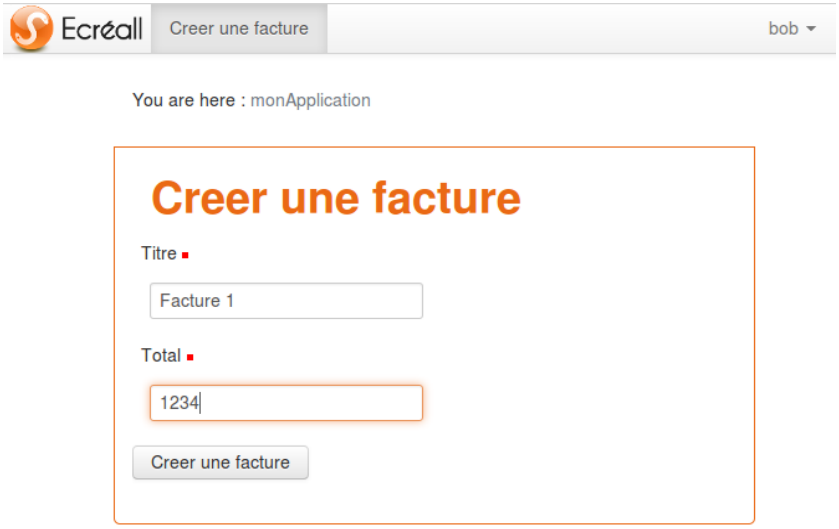


FIGURE 28: Formulaire de l'action **Creer une facture**

céder et effectuer les actions appropriées. La figure 31 montre les différentes actions qu'**Alice** peut effectuer sur la facture créée (**Refuser** ou **Payer**). Une seule de ces actions peut être effectuée puisque dans notre cas nous avons un choix exclusif entre ces deux actions.

6.2 L'APPROCHE MACOP

Dans cette section nous donnons une description générale de l'approche MACoP. Nous commençons par justifier notre choix concernant la réalisation d'un DSL (Domain Specific Language). Ensuite nous donnons notre vision des portails collaboratifs que nous considérons comme une application web centrée sur les processus métier

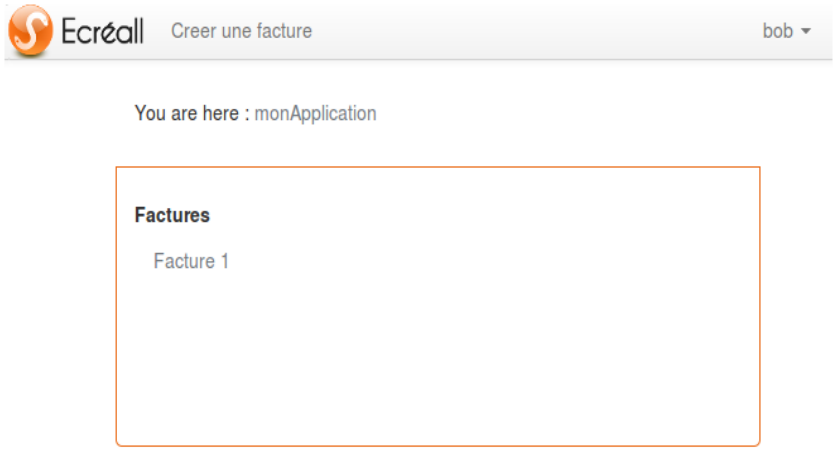


FIGURE 29: L'application vue par le **Vendeur**

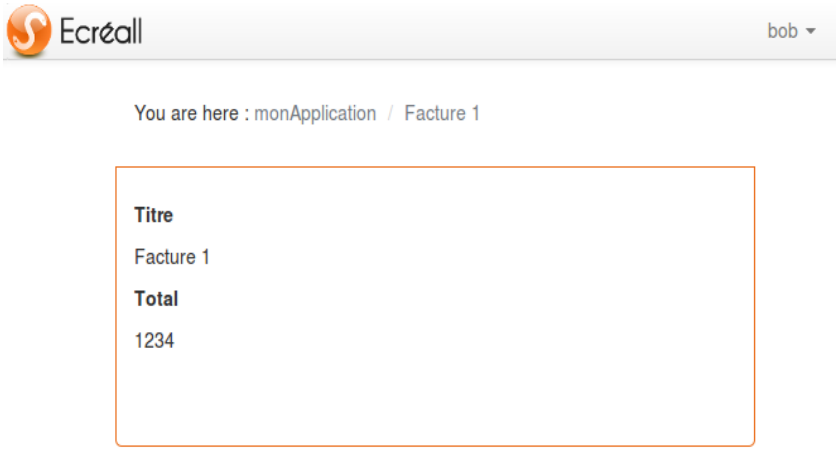


FIGURE 30: La facture créée vue par le **Vendeur**

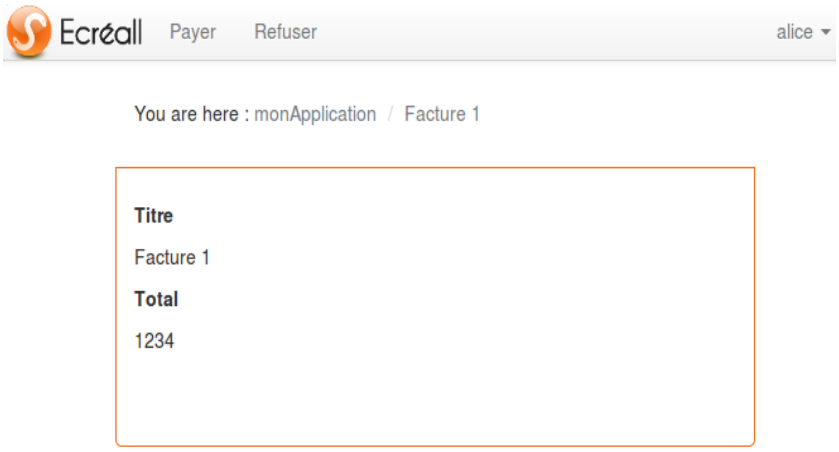


FIGURE 31: La facture créée vue par le **Client**

centrés sur les données. Enfin, nous donnons une description générale du métamodèle MACoP.

6.2.1 *Pourquoi un DSL ?*

La question d'utiliser un langage spécifique (DSL) ou un langage unifié (UML) est importante. Dans la section 5.2, nous avons indiqué la difficulté à mesurer l'ergonomie d'un métamodèle de façon non relative. En conséquence, nous avons choisi de privilégier l'accessibilité du langage pour les acteurs métier, entre autres par ce que nous considérons que le domaine de l'informatique est au service des autres domaines.

Le domaine de l'informatique a comme rôle principal de permettre à l'homme de contourner certaines contraintes du monde réel. Cela en proposant un environnement numérique permettant de simuler et/ou compléter la réalité. Afin de créer cet environnement, l'homme doit communiquer avec la machine en utilisant des langages qui sont généralement accessibles seulement aux informaticiens. Pour produire une application informatique, la réponse aux besoins métier réels doit être transformée en artéfact écrit en un langage informatique. Cette opération peut avoir un coût considérable en temps et en argent, ce qu'une entreprise peut difficilement se permettre dans un monde où son métier devient souvent évolutif et réactif. Afin de réduire les coûts de développement, une solution consiste à permettre aux entreprises de passer directement de l'expression de leurs besoins métier à l'application. Pour cela, la machine doit être capable de transformer automatiquement le langage métier, accessible par les acteurs métier, en un langage technique puis technologique. Nous avons écarté l'utilisation d'UML car, selon nous et par son origine, ce dernier est principalement dédié aux informaticiens et peut constituer une source d'ambiguïté pour les acteurs métier.

UML a d'abord été conçu comme un DSL graphique pour l'architecture des applications informatiques orientées objet puis des systèmes d'informations.

Dans un monde où la réactivité devient de plus en plus nécessaire, l'accessibilité du métamodèle décrivant les applications web par les non-informaticiens devient essentielle. Assurément, c'est à l'informatique de s'adapter aux autres domaines et non l'inverse. UML est l'une des tentatives les plus connues visant à unifier le plus grand nombre de domaines d'activités. L'usage d'UML2 ne garantit pas l'unicité d'interprétation des modèles, car le domaine d'activité du lecteur influence sa compréhension. Pour être moins ambigus, les modèles doivent être plus détaillés, mais du coup, ils deviennent plus difficiles à appréhender pour les acteurs métier ce qui augmente le risque d'erreurs d'interprétations.

UML propose de lever les ambiguïtés par l'utilisation de *profil* [49]. Ce qui permet d'adapter UML à un domaine donné. Cette adaptation a des limites, car le résultat dépend fortement de la syntaxe abstraite du métamodèle UML.

Un acteur métier sera plus à même de modéliser son métier dans un langage dédié à son domaine. Un tel langage est plus accessible et donc plus ergonomique qu'une modélisation unifiée. L'utilisation

Une ontologie, en informatique, est un modèle décrivant l'ensemble des termes et concepts propre à un domaine de connaissance donné.
Le jargon est le langage, ou le parler, propre aux acteurs d'un domaine donné.

d'un langage spécifique au métier d'un acteur est un choix permettant de refléter son ontologie (« son jargon ») , l'acteur y reconnait presque immédiatement une partie de son lexique et de sa sémantique, et n'est pas surpris par l'articulation des éléments imposés par le DSL. Nous rejoignons ainsi notre impératif d'accessibilité et d'expressivité. Sachant que l'objectif de notre démarche est la génération des portails collaboratifs à partir d'une modélisation accessible par les acteurs métier, le choix de la modélisation des systèmes d'information collaboratifs nous parait le plus pertinent. C'est pourquoi nous avons créé un DSL basé sur des métamodèles standardisés pour décrire nos modèles métier de nos portails collaboratifs. Nous ne proposons pas un DSL pour chaque métier, mais un DSL spécifique à la modélisation des processus métiers collaboratifs.

6.2.2 *Notre vision du portail collaboratif*

À partir d'un modèle décrivant le portail collaboratif à un niveau métier nous voulons générer le portail collaboratif associé. Ce portail collaboratif est une application web centrée sur les données permettant l'exécution des processus métier. Dans une telle application, les données (artéfacts métier ou objets de collaboration) doivent être, selon nous, les éléments centraux. L'utilisateur manipule des instances d'objets de collaboration en fonction des processus métier définis. Ce qui signifie que l'exécution des processus est implicite (déclenchée par le système) et centrée sur les données ce qui n'est pas le cas pour les solutions BPM d'orchestration des processus métier. Dans ces solutions, l'exécution des processus métier est explicite (déclenchée par l'utilisateur) et centrée sur les tâches comme vue dans la section 4.3.3.

Dans notre proposition, une application web centrée sur les données doit permettre à l'utilisateur de se déplacer d'une instance d'objet de collaboration à une autre. L'instance d'objet de collaboration est considérée comme le contexte principal. Selon les droits de l'utilisateur, il lui est permis d'exécuter un ensemble d'actions associées au contexte principal. Par exemple dans une application de gestion de factures, un client est amené à payer ses factures. Les factures sont les instances de l'objet de collaboration **Facture**, ou contextes principaux, et l'action de paiement est associée aux factures. Dans une telle application, le client navigue entre les factures et effectue l'ensemble des actions associées à la facture selon ses droits.

Les objets de collaboration sont reliés entre eux par des liens structuraux (voir figure 32). Par exemple un dossier contient des factures. Les droits de l'utilisateur déterminent la visibilité des liens. Une instance d'objet de collaboration peut avoir plusieurs vues qui lui sont associées (vues pour présenter l'objet, vues pour présenter les actions, etc.). Certains liens permettent à un utilisateur donné de changer de vue sans changer de contexte principal (voir figure 32). Les vues as-

sociées à un objet de collaboration peuvent proposer des liens sur d'autres objets de collaboration. De plus, ces liens ne correspondent pas toujours aux liens structuraux entre les objets de collaborations. Une vue est un ensemble de structures de données visuelles qui peuvent être composées de formulaires ainsi que des données propres à l'objet de collaboration ou de données associées à d'autres objets de collaborations. Dans notre proposition nous ne prenons pas encore en compte les aspects graphiques que nous appelons "le style" (c'est-à-dire la position dans la page, les couleurs, etc.). Une page web dans nos portails collaboratifs est composée de plusieurs vues.

En résumé, une application web peut être considérée comme un graphe dont les nœuds sont les pages web et les arcs les liens de navigation, alors que nos portails collaboratifs sont vus comme des graphes, dont les nœuds sont les instances des objets de collaboration et les arcs sont les actions déclenchable à partir de ces instances. La figure 32 montre la superposition de ces deux graphes avec leurs arcs de navigation respectifs.

Nous faisons la différence entre l'instance d'objet de collaboration (considérée comme le contexte principal) et ses différentes vues (qui servent à afficher des données visuelles). Un utilisateur, dans ce cas, navigue entre les différentes vues associées aux instances des objets de collaborations selon les contraintes dictées par les exigences métier.

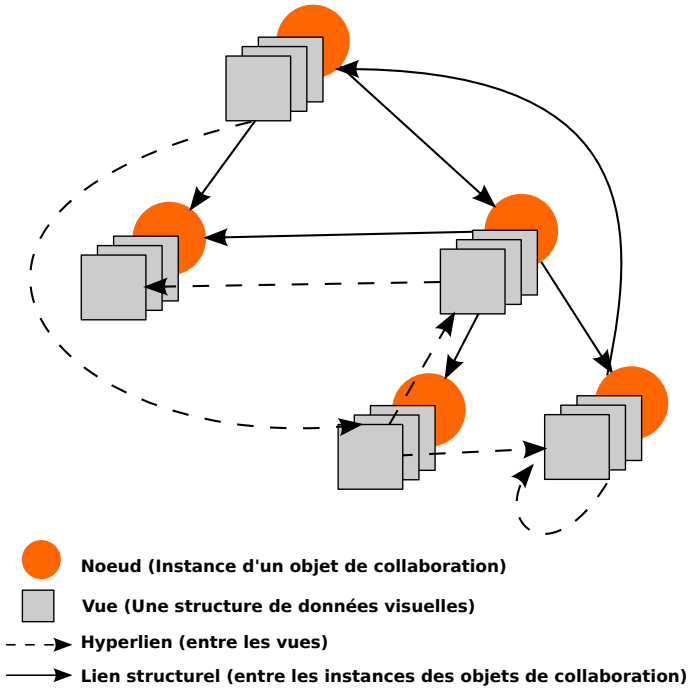


FIGURE 32: Superposition du graphe d'une application web sur le graph des instances des objets de collaborations

Ces graphes sont dynamiques et évoluent selon les actions effectuées sur les instances des objets de collaborations (création, modifi-

cation, etc.). Dans une collaboration et par rapport à un utilisateur donné, l'évolution de ces graphes (c'est-à-dire la dynamique de l'application) est concrétisée par l'enchaînement des actions métier contextualisé par des règles métier, par exemple l'attribution des rôles dans la collaboration. En effet, le fait de changer de rôle permet d'enlever ou ajouter des droits d'agir ou d'accéder aux données. En conséquence la perception de l'utilisateur évolue. Les utilisateurs, alors, peuvent voir l'application à l'exécution différemment selon les contraintes et les données de l'application.

Comme décrit dans la section 4.3.2, la description d'une application en se basant sur les liens hypertextes n'est pas, selon nous, adaptée. Nous proposons que les liens fonctionnels et structurels soient exprimés d'une manière factorisée à travers la description des objets de collaboration et les actions des processus métier. Cette présentation est, selon nous, la solution la plus raisonnable en terme de taille et par conséquent en terme de compréhension du modèle. La question alors posée est : comment abstraire les concepts techniques, par exemple la navigation, en besoin métier ?

Généralement, dans un portail collaboratif, nous pouvons identifier deux types de lien de navigation : la navigation structurelle et la navigation fonctionnelle.

- La navigation structurelle permet de naviguer entre les différentes vues informatives. Cette navigation est sans conséquence sur les données de l'application et est à but informatif. Elle permet de se déplacer dans le graphe des instances des objets de collaborations de l'application selon les droits d'accès de l'utilisateur.
- La navigation fonctionnelle permet de naviguer principalement entre les vues d'interaction (c'est-à-dire une vue demandant à l'utilisateur de saisir des données, par exemple un formulaire) afin d'effectuer un traitement sur les instances des objets de l'application. Une action du processus métier est un traitement à effectuer sur un ensemble de données. Ces données sont passées en paramètres pour l'action. En ce qui concerne la navigation fonctionnelle, c'est l'association de l'action à une instance d'objet de collaboration qui doit être spécifiée. L'instance est alors considérée comme le contexte principal par rapport à l'action.

Dans un portail collaboratif, la navigation est contextualisée selon les différentes contraintes métier comme l'attribution des rôles ou les contraintes de sécurité.

En considérant la visualisation des données comme une action, nous pouvons conclure que les liens de navigation (structurels et fonctionnels) dans un portail collaboratif sont tous des actions permettant de renvoyer une structure de données sur un contexte principal. De ce fait, nous pouvons abstraire la notion de lien de navigation en action métier contextualisée. *Une application collaborative est donc un enchaînement de flux de données contrôlé par les actions métier.*

Selon nous, la description de l'enchaînement des actions ainsi que la description des structures des données montrées par ces actions sont suffisantes pour capter les besoins métier du portail collaboratif à un haut niveau d'abstraction.

6.2.3 *Description générale de notre démarche*

Le but de nos travaux est la modélisation du portail collaboratif, basé sur le web, du système informatique à un niveau accessible par les acteurs métier. Nous nous sommes inspirés de ce qui a été fait dans le domaine de la modélisation des entreprises et plus spécifiquement du métamodèle **CIM-OSA** [74] et celui de **UEML** [7]. Ainsi nous avons identifié la question principale posée par ces métamodèles et qui consiste à identifier « *comment* le *qui* agit sur le *quoi*» ? Dans cette question, nous nous intéressons au *comment*, à *qui* et à *sur quoi*. Dans UEML, cette question est présente à travers la description des rôles et des objets de l'entreprise nécessaires aux rôles afin d'effectuer une activité donnée. Cette notion de nécessité inclut, pour UEML, le traitement et l'accès aux objets. Cette question a été uniformisée dans la norme **ENV 40003** [57] en proposant le point de vue fonctionnel pour les processus métiers (ici le « Comment ? »), le point de vue informationnel pour les objets de l'entreprise (ici le « Quoi ? ») et enfin le point de vue de ressources pour les rôles des acteurs de l'entreprise (ici le « Qui ? »). Dans cette norme, les rôles sont vus comme des ressources.

Nos travaux ont abouti au métamodèle baptisé MACoP pour « Modeling and Analysis of Collaborative Portals ». MACoP permet de modéliser les portails collaboratifs à un haut niveau d'abstraction, davantage fondé sur les concepts métier des systèmes d'information que sur les concepts techniques du web. Dans notre approche les détails techniques et les concepts web sont rajoutés automatiquement au modèle initial en appliquant des transformations qui ne se contentent pas de réaliser de simples transformations une à une, mais jouent le rôle d'expert, car elles contiennent l'expertise et les connaissances des maîtres d'œuvre et des développeurs web. Le métamodèle MACoP permet de spécifier uniquement les besoins de l'utilisateur et le cahier des charges de l'application.

Dans MACoP, nous avons divisé la modélisation des portails collaboratifs en différents points de vue traitant chacun d'un problème particulier. Pour cela, nous avons distingué le fond de la forme (voir figure 33) (c'est-à-dire le noyau fonctionnel (NF) de l'IHM). Notre approche consiste à métamodèliser les portails collaboratifs en se basant sur les notions du système d'information. Ainsi, nous avons séparé le fond en structures documentaires « Quoi ? », en rôles « Qui ? » et en comportements « Comment ? ». La forme, quant à elle est totalement séparée du fond. En effet, la forme décrit comment les ac-

teurs (« Qui ? ») perçoivent et interagissent avec les informations (« Quoi ? ») en suivant des actions métier (« Comment ? »).

Par exemple, dans une collaboration un **acteur** quelconque peut **déplacer** un **fichier** dans un **dossier** en mode **glisser-déposer**. Dans ce cas, l'**acteur** est le « Qui ? », les **dossiers** ainsi que les **fichiers** sont le « Quoi ? », l'action **déplacer** est le « Comment ? » enfin, le mode **glisser-déposer** présente le type d'interaction. Dans notre modélisation, la forme est séparée en : style statique « Comment voir le Quoi ? », style dynamique « Quand et comment voir le Quoi ? » et enfin interactions « Faire le Comment ? ».

L'imbrication successive des questions du QQOQCCP (voir section 5.3.1), nous permet de créer notre métamodèle en évitant les oublis, tout en respectant la notion des vues de modélisation de la norme **ENV 40003**.

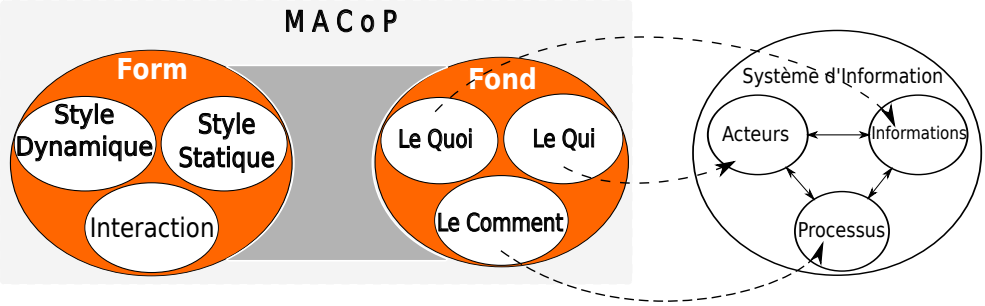


FIGURE 33: MACoP

Notre approche propose plusieurs niveaux de détails pour une même modélisation. Cela permet à un intervenant d'agir sur différents niveaux de détails selon sa compétence et ses rôles dans l'organisation.

Une modélisation simple permet d'avoir une application dans de brefs délais. Ce type de modélisation peut être concrétisé par la réutilisation de modèles. Dans notre approche nous avons introduit cet usage à travers la modélisation orientée aspect ainsi que la modélisation à base de composants réutilisables.

Dans la section suivante, nous décrivons principalement le métamodèle représentant le fond et permettant la modélisation du métier (l'ergonomie métier). Nous ne parlerons que brièvement de la modélisation du style de l'application (l'ergonomie du style) et de sa génération automatique. Ce dernier sujet sort du cadre de cette thèse et sera abordé dans les perspectives. Dans les autres sections nous détaillons les différentes extensions apportées au métamodèle MACoP afin de décrire les processus métier centrés sur les données. Enfin, nous décrivons la réutilisation de modèle dans MACoP.

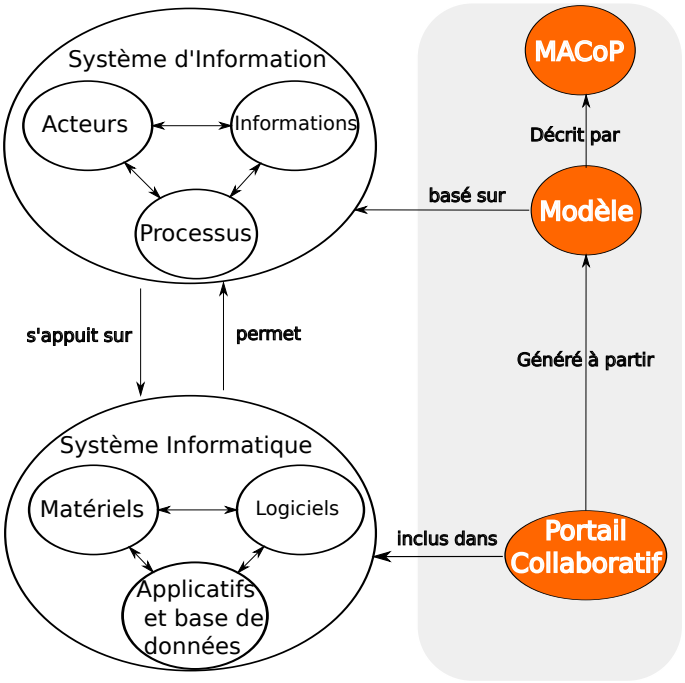


FIGURE 34: MAcoP et SI

6.3 MODÉLISATION DU MÉTIER

Nous proposons de modéliser le métier dans trois vues : la vue informationnelle qui concerne les structures documentaires et qui permet de répondre au « Quoi ? » la vue des ressources qui concerne les rôles et qui répond au « Qui ? » et enfin la vue fonctionnelle qui concerne le comportement et qui répond au « Comment ? ». Dans notre approche, ces trois vues suffisent à capter tous les besoins nécessaires pour le développement d'un portail collaboratif. Les autres questions du QQOQCCP sont appliquées dans chacune des vues, nous permettant ainsi de les compléter.

6.3.1 *Vue informationnelle*

Dans cette vue nous décrivons les objets de collaboration qui sont manipulés sur le portail collaboratif.

Cette vue permet aussi de spécifier des instances préinitialisées des objets de collaborations. Ces instances permettent de décrire les données initiales requises par le métier. Nous avons ainsi deux niveaux de modélisation : le niveau *modélisation de la structure de l'application* permettant de déclarer les objets de collaboration et *le niveau configuration de l'application* permettant la préinitialisation de ces objets. Dans ce qui suit, nous détaillons ces deux niveaux.

Modélisation de la structure de l'application

La modélisation de la structure de l'application permet de décrire les différents patrons utilisés dans une entreprise. Un patron est une déclaration de la structure qu'une donnée doit respecter. Selon nous, dans notre contexte, un patron pour une donnée est ce qu'est un métamodèle pour un modèle par rapport à la classification de modèle de l'OMG. Par exemple, dans une entreprise nous pouvons trouver le patron d'une facture qui décrit, d'une manière abstraite, comment une facture doit être. Le patron de la facture contient la description des différents attributs qu'une facture doit remplir afin d'être conforme aux exigences métier. Dans ce patron, nous pouvons trouver comme attribut le nom d'une facture qui est une chaine de caractère. Une facture dans ce cas doit avoir un nom.

Dans notre approche les patrons sont décrits par le concept *Objet de Collaboration*. Comme illustré à la figure 35, le concept d'*Objet de Collaboration* se spécialise en plusieurs concepts qui sont : **Application**, **Information**, **Profil** et **Processus Domaine**. Dans ce qui suit, nous détaillons ces concepts.

Dans le cadre de la modélisation des entreprises, le concept d'« Application » est considéré comme une ressource active (voir le métamodèle UEML [7] présenté dans la figure 4) pouvant être utilisée par un rôle dans le cadre d'une activité donnée.

La modélisation des entreprises permet la description du fonctionnement de l'entreprise dans sa globalité. Dans cette modélisation une application est vue comme une boite noire permettant de réaliser des traitements sur les informations de l'entreprise. Les détails de l'application, dans ce cas, ne sont pas décrits.

Notre approche consiste à modéliser une de ces applications (le portail collaboratif) dans le détail en se basant sur les besoins métier décrits dans le modèle de l'entreprise. Les autres types d'applications ne sont pas pris en compte et sont considérés comme des acteurs machine jouant un rôle dans la collaboration.

Au contraire de la modélisation des entreprises, dans la modélisation des systèmes d'information, la description de l'application est explicite. Comme nous l'avons vu au chapitre 2, la modélisation des systèmes d'information est une vue dématérialisée de l'entreprise. Par conséquent, la seule application identifiée dans notre modélisation est le portail collaboratif lui-même. C'est pourquoi le concept « Application » décrit la racine du portail collaboratif considérée comme un objet de collaboration.

Les **Information**s sont des objets de collaboration passifs qui permettent de décrire la structure métier de la collaboration (le plan documentaire).

Les **Profil**s sont des objets de collaboration passifs pouvant être des machines ou des humains. Ils sont des représentations informatiques des acteurs dans la collaboration. Un acteur est une entité active qui

peut être un humain ou une machine. Un acteur machine peut réaliser des actions d'une manière automatique selon sa fonction dans la collaboration. Un humain quant à lui peut déclencher des actions tout en interagissant avec l'application. Une interface homme-machine (IHM) permettant cette interaction est, donc, nécessaire pour ce type de profil.

Les **Processus Domaine** sont des objets de collaboration permettant de décrire les processus métier. Dans la vue informationnelle de notre approche, ce concept représente la vue passive (c'est-à-dire informationnelle) d'un processus métier. La vue fonctionnelle des **Processus Domaine** sera détaillée dans la section 6.3.3.

En résumé, les objets de collaboration suivants sont utilisés pour modéliser la structure de l'application :

- Les « **Information** » : objets de collaboration **sur lesquels on agit seulement**.
- Les « **Profil** » : objets de collaboration **qui agissent et sur lesquels on agit**.
- Les « **Processus Domaine** » : objets de collaboration **qui décrivent comment agir et sur lesquels on agit**.
- Les « **Application** » : objets de collaboration **qui permettent d'agir et sur lesquels on agit**.

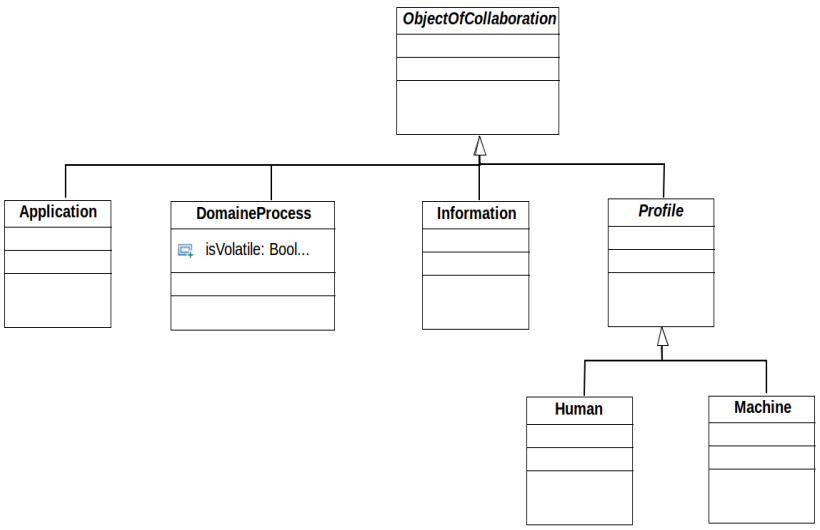


FIGURE 35: Le métamodèle des objets de collaboration

Pour faciliter la réutilisation et la composition, nous avons enrichi le concept d'objet de collaboration avec des notions d'héritage, de type abstrait, de composition, d'associations, de contraintes, etc. Pour cela, nous avons rendu le concept d'objet de collaboration (voir figure 36) équivalent au concept « Classifier » d'UML ou de MOF sans la partie comportementale, qui dans notre approche est décrite dans la vue fonctionnelle.

Les contraintes et les invariants sont exprimés dans le langage OCL [103] qui est un langage de contraintes et de requêtes standardisé situé à un niveau d'abstraction élevé et indépendant des détails technologiques.

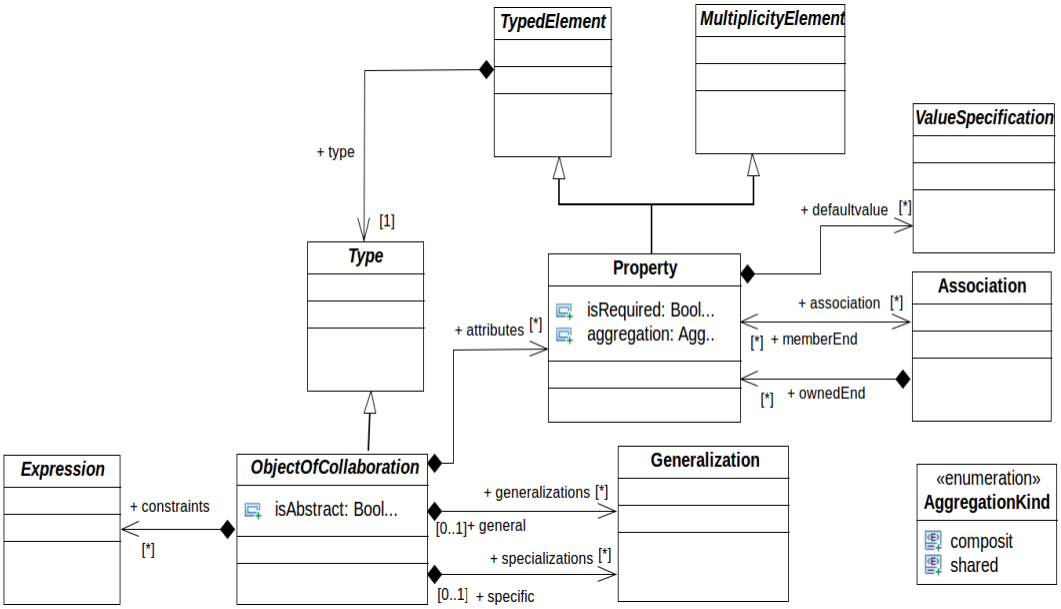


FIGURE 36: Une partie du coeur du métamodèle MACoP

Chaque objet de collaboration doit être associé à un ou plusieurs processus métier le manipulant (création, modification, etc.). Dans le cas où l'objet de collaboration n'est pas manipulé par aucun processus métier, un processus par défaut sera créé lors de la transformation pour pouvoir manipuler cet objet.

Configuration de l'application

À l'exécution le portail collaboratif est constitué d'un graphe dynamique dont les nœuds sont les instances des objets de collaboration. Dans la section précédente, nous avons décrit les objets de collaborations qui sont les patrons dans une entreprise. Dans notre approche une instance d'un objet de collaboration est une donnée de l'entreprise conforme à un patron.

Afin d'assurer le minimum requis par le métier, une instance de la structure des objets de collaboration peut être modélisée dans la vue informationnelle. Cette instance correspond au graphe des instances des objets de collaboration du portail collaboratif. Il est, donc, possible d'initialiser l'application en instanciant certains objets de collaboration. Comme illustré à la figure 37, MACoP dispose du concept « **InstanceSpecification** » permettant d'instancier un objet de collaboration d'une manière générale. Dans MACoP, nous trouvons, aussi, le concept « **ProfilInstanceSpecification** » qui sont des « **InstanceSpecification** » auxquelles on peut affecter des rôles. Les « **ProfilInstance-**

Specification » permettent d'instancier les objets de collaboration de type « **Profil** ».

L'instanciation d'un objet de collaboration peut être faite de deux manières : une instanciation désignant une instance, d'un portail collaboratif, bien identifiée d'un objet de collaboration et une instanciation désignant une des instances, d'un portail collaboratif, non identifiées d'un objet de collaboration. Dans le cas où l'instance est non identifiée, nous dirons que c'est une instance généralisée ou instance quelconque.

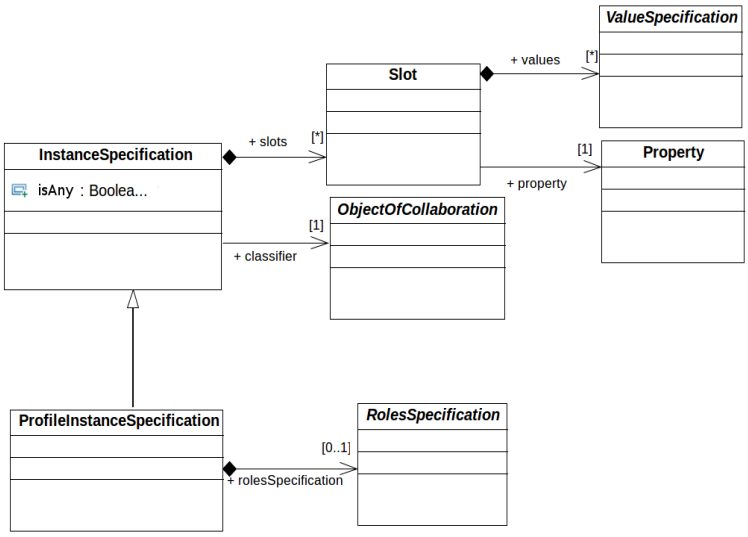


FIGURE 37: Le métamodèle des instances

Une « **InstanceSpecification** » désignant une instance bien identifiée, par exemple la facture d'un client donné connu, permet de désigner une instance connue dans le graphe des instances des objets de collaboration d'un portail collaboratif. Les informations associées à cette instance (c'est-à-dire ses relations avec les autres instances d'objets de collaboration) permettent de la retrouver. Dans MACoP, les « **InstanceSpecification** »**s** permettant de désigner des instances identifiées sont utilisées pour décrire la configuration de l'application. Dans cette configuration, chaque instance est identifiée et retrouvable dans le portail collaboratif par rapport aux autres instances. La figure 38 montre ce qu'une « **InstanceSpecification** » désigne dans un portail collaboratif représenté par son graphe d'instance d'objet de collaboration. L'instance désignée est retrouvable à partir d'une requête sur le graphe des instances du portail collaboratif.

En ce qui concerne les instances quelconques, quant à elles, ne sont pas connues dans le graphe des instances des objets de collaboration du portail collaboratif. Une « **InstanceSpecification** » généralisée permet de désigner une instance quelconque dans le portail collaboratif. Cette instance d'objet de collaboration est de même type et ayant la même signature que l'« **InstanceSpecification** » généralisée (c'est-

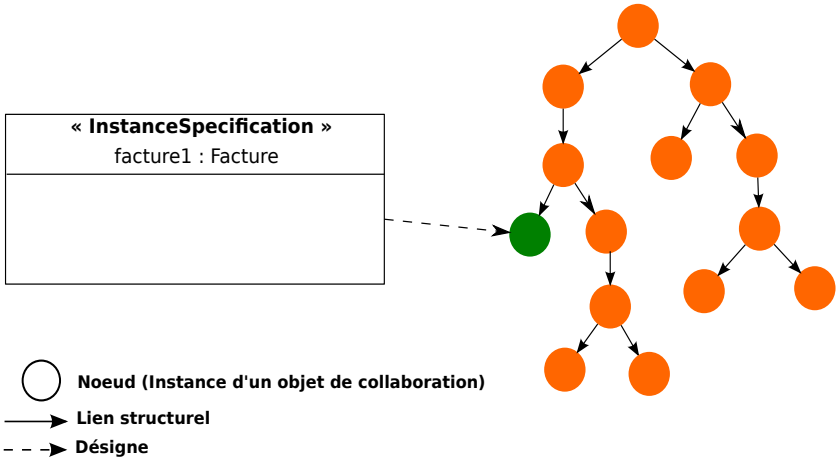


FIGURE 38: L'instance identifiée par rapport aux instances du portail collaboratif

à-dire possédant les mêmes valeurs des attributs), ce qui généralise les règles métier écrites. Les informations associées à cette instance (c'est-à-dire ses relations avec les autres instances d'objets de collaboration) ne permettent pas de la retrouver. Dans une collaboration, les acteurs peuvent être amenés à effectuer un ensemble d'actions sur des instances d'objets de collaboration quelconques, par exemple le fait de pouvoir créer un fichier dans un dossier quelconque. Dans MACoP, le concept « InstanceSpecification » contient un attribut booléen « isAny » permettant de spécifier si l'instance désignée est identifiée ou généralisée. La figure 39 montre ce qu'une « **InstanceSpecification** » généralisée désigne dans un portail collaboratif représenté par son graphe d'instance d'objet de collaboration. Cette « **InstanceSpecification** » désigne une des instances représentées en vert et non toutes les instances en vert. L'instance désignée n'est pas retrouvable à partir d'une requête sur le graphe des instances du portail collaboratif. Elle doit être spécifiée par l'utilisateur du portail collaboratif généré.

La description des relations entre les instances des objets de collaboration de l'application et les actions métier d'une manière précise est primordiale dans MACoP. Les « InstanceSpecification » identifiées permettent de spécifier une relation désignée avec l'action métier. Les « InstanceSpecification » généralisées, quant à elles, permettent de spécifier une relation non désignée avec l'action métier. Cette partie est décrite en détail dans la section 6.3.3.

6.3.2 *Vue des ressources*

Une fois les objets de collaboration (c'est-à-dire les types ou patrons) modélisés et l'application pré-initialisée (c'est-à-dire les instances), l'utilisateur modélise les rôles et leur hiérarchie dans la collaboration. À un rôle est attaché un ensemble d'activités confiées à des ac-

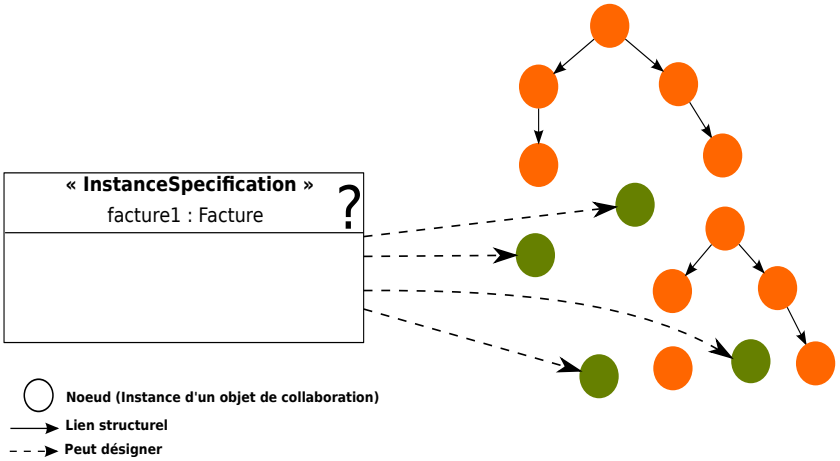


FIGURE 39: L'instance généralisée par rapport aux instances du portail collaboratif

teurs métier dans le cadre d'une responsabilité, ceci en fonction d'une qualification ou de compétences. C'est la « fonction » professionnelle au sein de l'entreprise qui donne le droit à cet acteur d'accéder et/ou d'agir sur les instances des objets de collaboration. Plusieurs acteurs peuvent tenir le même rôle. Et un acteur peut avoir plusieurs rôles. Les rôles sont les concepts clefs de l'organisation des individus au sein d'une collaboration.

Comme décrit dans la section 2.1, la vue des ressources fournit une représentation de l'ensemble des moyens nécessaires pour mettre en œuvre les activités. Cette représentation est centrée sur le côté fonctionnel et non sur le côté informationnel. Nous nous sommes inspirés du métamodèle EUML [7] qui limite cette représentation aux rôles avec leurs compétences et leurs qualifications. Nous avons, ainsi, une séparation entre le côté fonctionnel et le côté informationnel d'une entité active (c'est-à-dire un acteur humain ou un acteur machine dans une collaboration). Comme illustré à la figure 40, un acteur (humain ou machine) est représenté par une instance de profil « **ProfilInstanceSpecification** » (côté informationnel) et une affectation de rôles (côté fonctionnel).

Notre approche permet de modéliser les relations hiérarchiques entre rôles. De même, un rôle peut être spécifié comme abstrait pour permettre la factorisation par héritage d'un ensemble d'actions communes (voir figure 41).

MACoP dispose du concept abstrait de « **RoleSpecification** » afin d'affecter un rôle ou un ensemble de rôles à une instance de profil ou un ensemble de tâches dans une collaboration. L'affectation des rôles à une instance de profil signifie que l'utilisateur du portail collaboratif, ayant cette instance, se voit affecter un ensemble de "clés" (un rôle = une clé). L'affectation de rôles à une tâche (ou un ensemble de tâches), quant à elle, signifie que pour pouvoir effectuer cette tâche (ou l'ensemble de tâches), l'utilisateur du portail collaboratif

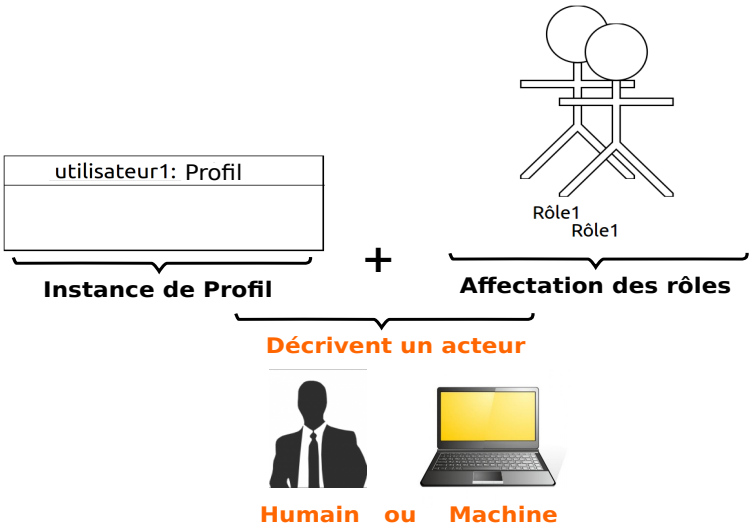


FIGURE 40: La description d'un acteur dans MACoP

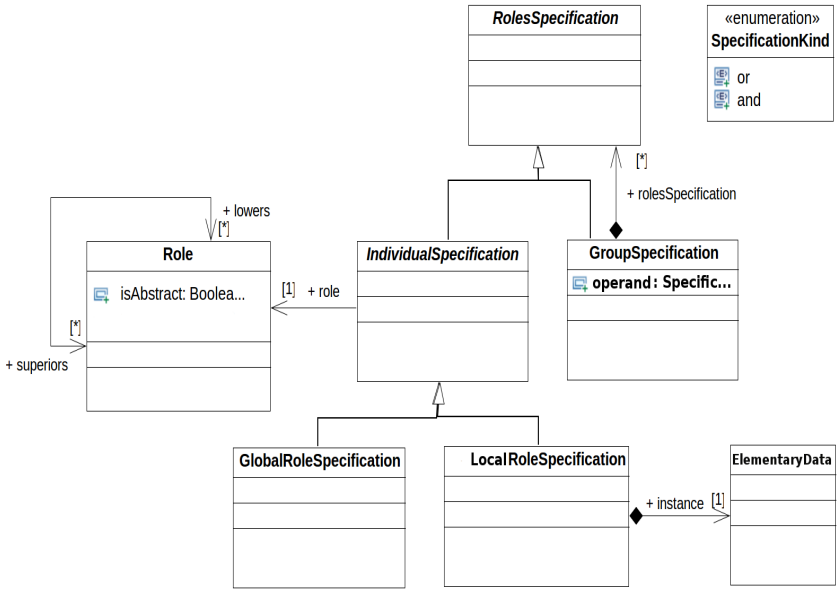


FIGURE 41: Le métamodèle des rôles

doit avoir toutes les clés correspondantes, c'est-à-dire qu'il doit être affecté des mêmes rôles.

La relation hiérarchique entre les rôles ainsi que la possibilité de spécifier des rôles abstraits dans le modèle permet de faciliter la maintenance et l'écriture d'actions communes, mais peut augmenter significativement le nombre de rôles. Cette solution n'est pas toujours fidèle au fonctionnement de l'entreprise à cause de l'utilisation des rôles abstraits.

Dans notre approche nous proposons d'affecter les rôles sous forme d'expression (expression de rôles) afin de minimiser le nombre de rôles utilisés tout en restant fidèle au fonctionnement de l'entreprise.

Par exemple « **Role1 and Role2** » est une expression de rôle. Le fait d'affecter cette expression à un ensemble de tâches signifie que l'utilisateur du portail collaboratif ayant le rôle **Role1** et le rôle **Role2** est autorisé d'effectuer ces tâches. MACoP dispose du concept « **GrouSpecification** » afin de décrire les expressions de rôles. Ce concept est détaillé dans la section suivante.

Comme décrit précédemment, le concept de rôle dans une collaboration est défini comme la « fonction » professionnelle d'un acteur qui lui donne le droit d'accéder et/ou d'agir sur les instances des objets de collaboration. Un rôle est associé à un champ d'action. Par exemple, le rôle Administrateur d'une application web a, en général, un champ d'action couvrant toutes les fonctionnalités de l'application permettant ainsi à l'utilisateur ayant ce rôle d'agir sur toutes les données de l'application sans restriction. Dans ce cas, le rôle Administrateur est affecté à un utilisateur de l'application web par rapport à toute l'application. Dans une collaboration, le champ d'action d'un rôle peut être réduit. Dans ce cas l'utilisateur ayant ce rôle ne peut agir sur les instances des objets de collaboration que dans le cadre de cette restriction. Cela signifie, aussi, que le rôle de l'utilisateur n'a de validité que dans le cadre de cette restriction. Par exemple, dans une équipe de développement de projet informatique on trouve plusieurs chefs d'équipe. Chaque chef d'équipe dirige un projet de développement bien défini. Dans ce cas, le chef d'équipe a un champ d'action restreint par rapport au projet qu'il dirige. Les autres chefs d'équipe n'ont pas les mêmes droits sur le projet que le chef d'équipe du projet. Dans notre approche, nous disons que l'affectation d'un rôle a une portée par rapport à la collaboration. Dans MACoP, la portée d'une affectation est définie par rapport aux instances des objets de collaboration. MACoP dispose du concept « **LocalRoleSpecification** » pour affecter des rôles avec une portée par rapport à un ensemble d'instances d'objets de collaboration et le concept « **GlobalRoleSpecification** » pour affecter des rôles avec une portée par rapport à toute la collaboration. Ces concepts sont des sous types du concept abstrait « **IndividualRoleSpecification** » permettant d'affecter un seul rôle à une instance de profil ou à un ensemble de tâches.

Un champ d'action est un espace dans lequel un acteur a une certaine influence lui permettant d'exercer des activités données.

Dans les sous-sections suivantes, nous détaillons l'affectation des rôles sous forme d'expression ainsi que la notion de la portée des affectations des rôles dans MACoP.

L'affectation des rôles sous forme d'expression

Dans une collaboration, on trouve souvent des rôles comme **Assistant** ou **Gérant** qui partagent les mêmes actions. Ce cas peut-être modélisé de deux façons : Une solution consiste à affecter les actions à chacun des deux rôles, ce qui constitue une « expression de rôles » : **Assistant ou Gérant**. Une autre solution consiste à abstraire les deux rôles en un troisième rôle commun, par exemple **Respons-**

able, auquel on affecte les actions communes. Les deux rôles concrets héritent alors du rôle abstrait, par exemple le rôle **Assistant** et le rôle **Gérant** héritent du rôle **Responsable**. Dans ce cas, un utilisateur ayant le rôle **Assistant** ou le rôle **Gérant** se voit aussi attribuer les tâches affectées au rôle **Responsable**. Une utilisation abusive de cette solution peut conduire à une augmentation considérable du nombre de rôles dans une application. La première solution basée sur les « expressions de rôles », quant à elle, est fidèle au fonctionnement de l'entreprise, mais rend la maintenance difficile (il ne faut pas oublier de maintenir conjointement les deux rôles).

Les deux solutions ont chacune leur avantage et leurs inconvénients. Dans MACoP le concepteur du portail collaboratif peut appliquer les deux solutions afin de tirer profit des avantages de chacune. L'affectation des rôles à une instance de profil ou à un ensemble de tâches dans notre proposition peut être individuelle « **IndividualSpecification** », dans ce cas un seul rôle est affecté, par exemple affecter le rôle **Responsable** à un utilisateur du portail collaboratif. Dans notre proposition, l'affectation peut être, aussi, en groupe « **GroupSpecification** » permettant de grouper un ensemble de « **RoleSpecification** ». L'affectation en groupe décrit une expression utilisant les opérateurs booléens exprimant une relation « **AND** » ou une relation « **OR** » entre les différentes « **RoleSpecification** » du groupe. Par exemple, pour affecter l'« expression de rôles » **Assistant ou Gérant** nous utilisant un « **GroupSpecification** » avec l'opérande **OR** et dans lequel nous avons un « **GlobalRoleSpecification** » référençant le rôle **Assistant** et un autre « **GlobalRoleSpecification** » référençant le rôle **Gérant**. Par composition le « **GroupSpecification** » permet de spécifier des expressions de rôles variées.

La portée de l'affectation d'un rôle

Afin d'expliquer la notion de portée des affectations de rôles, dans notre proposition, nous nous servons de l'exemple que nous avons décrit dans la section 6.1. Dans cet exemple, un utilisateur ayant le rôle **Vendeur** créer une facture. Ensuite un utilisateur ayant le rôle **Client** la paye ou la refuse. L'application générée permet l'inscription de plusieurs utilisateurs ayant le même rôle **Client** (nous les appelons clients). Les clients ont le droit de voir la liste des factures de l'application. Notre exemple donne la permission à tous les clients de refuser ou de payer une facture quelconque. Cela n'est pas correct puisqu'un client doit seulement payer ou refuser la facture qui lui est destinée. Par conséquent chaque client doit agir seulement sur ses factures. Il y a donc une relation entre un client et les factures. La question est, donc : comment modéliser cette relation ? Dans notre approche cette relation peut être soit une relation d'ordre informationnelle soit une relation d'ordre fonctionnelle (c'est-à-dire la fonction professionnelle).

La relation d'ordre informationnelle est modélisée dans la vue informationnelle par une association. Par exemple, l'association multiple « mesFactures » entre le profil **Utilisateur** et l'objet de collaboration **Facture** (voir figure 42). Un client (une instance du profil **Utilisateur** avec le rôle **Client**), dans ce cas, peut référencer sa liste des factures.

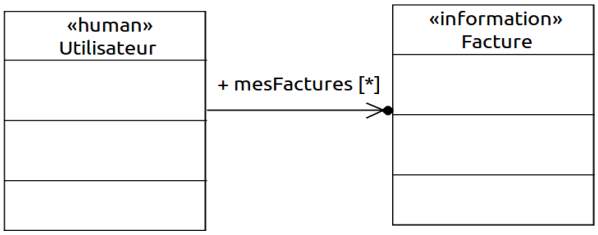


FIGURE 42: La relation « mesFactures » d'ordre informationnelle entre l'Utilisateur et la Facture

La relation d'ordre fonctionnelle, quant à elle, est une relation désignée par la « fonction » professionnelle de l'utilisateur sur une instance de facture bien définie. Cette « fonction » professionnelle n'est valide que sur un certain nombre de factures bien définies (dans notre exemple les factures du client). Cela signifie que l'utilisateur ne peut agir sur une facture que s'il possède une « fonction » professionnelle sur cette facture. Une « fonction » professionnelle induit, donc, une restriction ou une portée par rapport à un ensemble d'instances d'objet de collaboration.

Sachant que le rôle est la « fonction » professionnelle d'un acteur dans une collaboration, nous avons rendu possible la spécification de la portée d'une affectation d'un rôle dans MACoP. Dans ce cas, la portée d'une affectation d'un rôle peut être restreinte (c'est-à-dire locale) à une instance d'objet de collaboration, par exemple le bénéficiaire d'une facture (ici **Bénéficiaire** est le rôle et **facture** est l'instance de l'objet de collaboration **Facture**). Afin de spécifier une affectation locale de rôle, MACoP dispose du concept **« LocalRoleSpecification »** dans lequel il faut spécifier le rôle à appliquer d'une manière locale ainsi que les instances des objets de collaboration sur lesquelles cette affectation s'applique. Le rôle dans ce cas n'est valide que si l'utilisateur agit sur les instances spécifiées. La figure 43 montre une affectation locale du rôle **Bénéficiaire** à l'utilisateur **utilisateur1** par rapport à l'ensemble des factures **{facture1...factureN}**. L'utilisateur1 est donc le bénéficiaire de cet ensemble.

MACoP dispose aussi du concept **« GlobalRoleSpecification »** permettant d'affecter le rôle d'une manière globale par rapport à toute l'application.

Dans MACoP, il y a une distinction entre le rôle et l'affectation des rôles dans l'application. Avec cette distinction, un même rôle peut être affecté de manière locale ou globale. Dans notre proposition, une affectation locale d'un rôle par rapport à une instance d'objet de col-

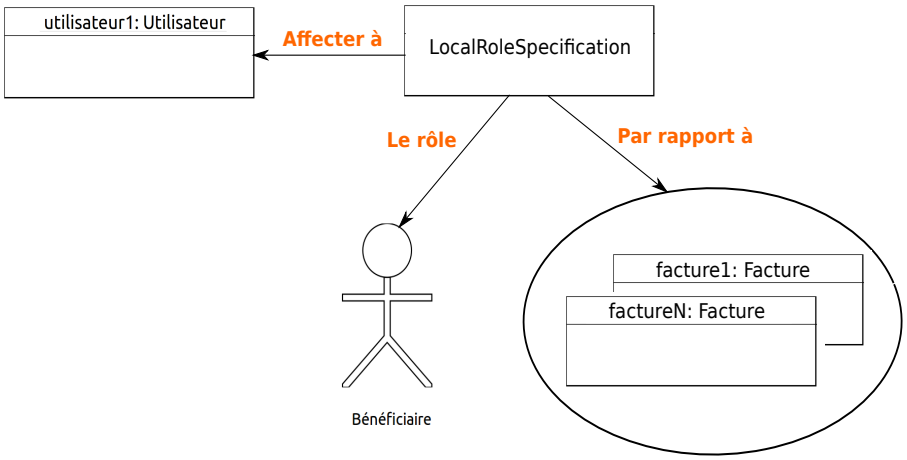


FIGURE 43: L'affectation locale du rôle **Bénéficiaire** par rapport à un ensemble de factures

laboration devient globale dès que l'instance d'objet de collaboration est considérée comme une généralisation (c'est-à-dire une « **InstanceSpecification** » généralisée). Par exemple le rôle propriétaire de la voiture immatriculée TRUC est un rôle affecté d'une manière locale par rapport à cette voiture, et le rôle propriétaire d'une voiture est un rôle affecté d'une manière locale par rapport à une voiture quelconque, ce qui est équivalent à une affectation globale de ce rôle.

6.3.3 *Vue fonctionnelle*

Cette vue décrit la dynamique de l'application exprimée sous forme de processus métier. Ces processus explicitent l'enchaînement logique des actions. Ces actions sont associées à des spécifications de rôle (voir la section 6.3.2). Une action est un ensemble d'opérations élémentaires à effectuer sur les données de l'application. Comme montrée dans la figure 44, notre modélisation propose plusieurs niveaux de modélisation afin de spécifier le fonctionnement de la collaboration. Nous avons le niveau « **Domaine** » décrivant les domaines d'activités. Chaque domaine d'activité contient une vue simplifiée, appelée ici « **Processus Domaine** », des processus métier. Puis les « **Processus Domaine** ») sont eux-mêmes détaillés dans le niveau inférieur suivant, afin de faire apparaître les détails du processus métier. Ces processus métier peuvent être hiérarchiques afin d'exprimer le métier de la collaboration d'un niveau moins détaillé à un niveau plus détaillé. Dans notre approche la hiérarchisation des processus métier est concrétisée par l'utilisation des sous-processus (non montré sur la figure) ainsi que de processus imbriqués (imbrication de processus métier). Enfin, les processus métier sont eux-mêmes composés d'actions qui peuvent être elles aussi, détaillées en composants. Ceci se fait dans le niveau le plus bas sur la figure. Ce niveau permet de décrire des com-

posants utilisés pour détailler le comportement (ou *implémentation*) d'une action, ainsi que des services, qui représentent des fonctionnalités comme l'impression (non montré sur la figure).

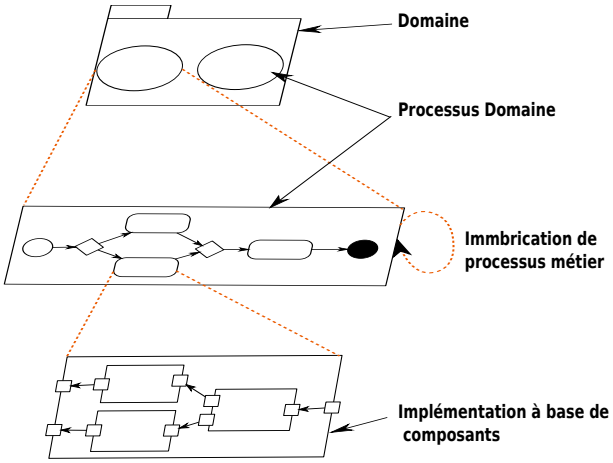


FIGURE 44: Décomposition des comportements dans MACoP

Afin de modéliser nos processus métier nous avons préférer l'utilisation d'un standard mis en place par des acteurs métier. Notre choix s'est orienté vers la notation et le métamodèle BPMN 2.0.

Dans cette section, nous commençons par justifier notre choix concernant le formalisme BPMN ainsi que les limites de ce formalisme. Ces limites ne nous permettent pas de générer le code complet de nos portails collaboratifs. De ce fait, nous avons complété la modélisation des processus métier. Le reste de cette section décrit les différentes extensions apportées afin de permettre la génération complète du code. Pour cela nous décrivons la relation que nos processus métier peuvent avoir avec les instances des objets de collaboration qu'ils manipulent. Puis nous montrons comment les données du portail collaboratif sont décrites dans MACoP et nous détaillons leur utilisation pour les contextes d'action des processus métier. Après, nous expliquons la notion de processus imbriqués et leur différence avec les sous- processus. Ensuite, nous parlons de l'implémentation des actions à base de composants et enfin nous détaillons brièvement le concept de service dans MACoP ainsi que son utilisation.

Les processus métiers

Notre proposition permet de regrouper les processus métier modélisés par domaine d'activité. Cela permet d'organiser le modèle pour une meilleure lisibilité. Nous nous sommes inspirés du métamodèle **CIM-OSA** pour introduire le concept de domaine d'activité, « **Domaine** » dans notre métamodèle.

Les processus métiers [51] permettent de décrire comment les données de l'application seront manipulées par les acteurs. Ils peuvent être représentés par des diagrammes d'activité d'UML, ou à l'aide

de diagramme BPMN 2.0. Les deux approches sont standardisées par l'OMG [100]. Plusieurs études montrent que BPMN a une représentation plus expressive et plus abstraite pour les processus métier que le diagramme d'activité d'UML. Les travaux de [40] présentent une étude comparative entre le diagramme d'activité d'UML et le BPMN 1.1. Nous pouvons également citer [85] qui ont proposé une transformation de BPMN vers le diagramme d'activité d'UML et qui ont rencontré des problèmes dus à la richesse de BPMN par rapport au diagramme d'activité d'UML. Dans notre proposition, nous avons écarté l'utilisation du diagramme d'activité d'UML peu adapté à la modélisation des processus métier, et nous avons retenu BPMN 2.0 qui est plus expressif.

Dans BPMN, un processus métier *Process* est représenté par un graphe composé de nœuds et de connexions entre ces nœuds. Les nœuds peuvent être des actions des événements (comme l'événement de début du processus ou l'événement de fin du processus) ou des nœuds de branchement permettant de contrôler le flux d'exécution des actions. Les connexions, appelées flux de séquences, permettent de décrire le flux de contrôle de l'exécution des nœuds (c'est-à-dire l'enchainement des nœuds). Des contraintes peuvent être spécifiées sur ces flux. Cela permet de bloquer le flux d'exécution au niveau du flux de séquence si la contrainte du flux de séquence est évaluée à faut. Dans BPMN, les nœuds du processus peuvent être regroupés en *Lane*. Un *Lane* désigne un responsable ou un collaborateur, par exemple un acteur, une unité organisationnelle, etc. Les actions d'un *Lane* sont affectées à ce responsable. Dans l'annexe A nous avons illustré les concepts principaux de BPMN.

Malgré sa puissance de représentation, BPMN présente une notation standardisée permettant d'exprimer les processus métier dans le but de les étudier, les analyser ou les simuler et non dans le but de les exécuter.

BPMN reste incomplet pour la génération complète de code et imprécis sur certains points. Par exemple, dans BPMN les contraintes associées à l'exécution du processus métier sont spécifiées au niveau des flux de séquences. De ce fait, il n'y a qu'un seul niveau de sécurité ce qui est, selon nous, insuffisant. En effet, dans BPMN, si nous voulons bloquer l'exécution d'un processus métier pour des raisons de sécurité représentées par une contrainte, il faut écrire la contrainte sur tous les flux de séquences du processus. Cela n'est pas raisonnable et peut être évité en éclatant les contraintes sur plusieurs niveaux (processus, *lanes*, flux de séquences). Afin de simplifier la modélisation et d'avoir un système de sécurité plus solide, nous proposons la possibilité de spécifier des contraintes sur plusieurs niveaux de sécurité par rapport au processus (voir figure 45). Le premier niveau de sécurité s'applique sur le processus d'une manière générale. Par exemple, afin d'exécuter un processus, le profil de l'utilisateur doit

être dans l'état actif. Le deuxième niveau de sécurité s'applique sur les *Lanes* et enfin le troisième niveau de sécurité s'applique sur les flux de séquence. Afin qu'un utilisateur de l'application puisse exécuter une action, toutes les contraintes de tous les niveaux de sécurité doivent être respectées.

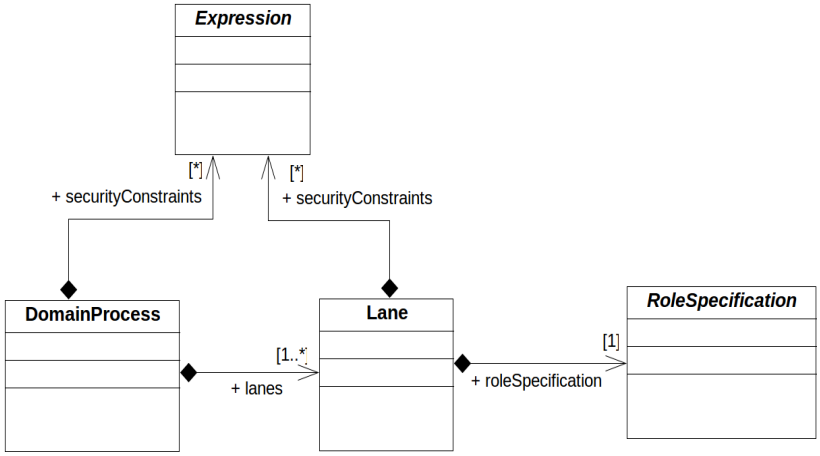


FIGURE 45: Une partie du métamodèle des processus métier

D'autres manques existent dans BPMN qui ne nous permettent pas de décrire d'une manière précise les actions métiers, comme sur quoi s'applique l'action ? Que produit-elle et où le produit-elle dans l'application ? Qu'a besoin d'étudier l'utilisateur afin de l'exécuter ? Quelles sont les données dont l'action a besoin afin d'être exécutée ? Ou, quel est le comportement exact d'une action ? Ces questions nous ont permis de détailler la question « Comment ? » située à un niveau de détails plus élevé. La réponse à ces questions nous permet d'avoir un modèle à partir duquel nous générons le code complet et opérationnel de nos portails collaboratifs. Ces questions portent essentiellement sur les instances des objets de collaboration manipulées par les processus métier.

Pour répondre à ces questions nous devons étudier les types de relations qu'une instance d'objet de collaboration peut avoir avec un processus métier.

Les relations entre les objets de collaboration et les processus métier

Dans notre approche, la modélisation des processus correspond à décrire leurs définitions. Dans l'application générée, une définition d'un processus peut être instanciée pour avoir *in fine* plusieurs instances d'une même définition de processus. Une définition de processus métier décrit, d'une manière générale, l'enchainement des actions à appliquer sur les instances des objets de collaboration. Ces instances peuvent être de même type ou de types différents. Un processus doit pouvoir désigner ses instances qu'il manipule. Cette désignation des instances des objets de collaboration est nécessaire pour

la récupération de ces instances. Cette récupération est indispensable pour l'évaluation des différentes contraintes (spécifications des rôles, contraintes sur les flux de séquences, contraintes de sécurité, etc.) définies dans le processus métier.

Nous distinguons deux types de relations entre un processus et une instance d'objet de collaboration : les relations désignées et les relations non désignées.

- Une relation désignée : l'instance de l'objet de collaboration est soit créée par le processus, soit choisie par l'utilisateur, ou par le système pendant l'exécution. Dans le cas où l'instance est créée par le processus, elle est désignée tout au long du processus à partir de sa création (passage de l'état non existant à l'état existant désigné). Dans le cas où l'instance est choisie par l'utilisateur pendant l'exécution, elle est désignée tout au long du processus à partir de sa spécification (passage de l'état existant non désigné à l'état existant désigné). Les contraintes associées au processus ne doivent pas porter sur l'instance avant sa désignation. Ce type de relation représente une association simple et implicite (c'est-à-dire non présente dans le modèle) entre le processus et l'instance de l'objet de collaboration. Par exemple, dans le domaine de gestion des factures on peut avoir un processus permettant de créer un **devis** dans un **projet** quelconque. Ensuite, l'utilisateur va pouvoir valider le **devis** créé. Une fois le **devis** validé, l'utilisateur va pouvoir créer la **facture** à partir des données du **devis**. Afin de créer la **facture**, le processus doit pouvoir récupérer le **devis** en question. Cela nécessite que le devis soit désigné par le processus. Le fait que le **devis** soit créé par le processus fait que le **devis** est désigné par le processus.
- Une relation non désignée : dans ce cas, l'instance de l'objet de collaboration est quelconque (c'est-à-dire généralisée). Elle n'est pas désignée et doit être choisie à l'exécution par l'utilisateur du portail collaboratif. La désignation est alors instantanée et est valide le long de l'action et non du processus. Ici, les contraintes associées au processus ne doivent pas porter sur l'instance avant sa spécification. Par exemple, dans une collaboration on peut avoir des processus de gestion de fichiers donnant aux utilisateurs la possibilité de déplacer un **fichier** quelconque dans un **dossier** quelconque. Dans ce cas, les **fichiers** et les **dossiers** sont non désignés par rapport au processus. À chaque exécution de l'action de déplacement, le dossier ainsi que le fichier à déplacer doivent être choisis par l'utilisateur du portail collaboratif.

La relation désignée et la relation non désignée font partie de la sémantique du métamodèle MACoP. Ces deux relations ne sont pas présentes explicitement dans ce métamodèle.

La description des données dans les processus métier

Une donnée est un concept général qui peut décrire une instance d'un objet de collaboration ou une valeur du paramètre d'une instance d'objet de collaboration. Une donnée peut être aussi le résultat d'une requête sur l'ensemble des objets de l'application.

Dans MACoP, les actions métier manipulent des données. Il convient, donc, de définir comment les données manipulées par ces actions sont décrites dans MACoP. Sachant que notre modélisation consiste à modéliser les flux des données contrôlés par les actions métiers, nous pensons qu'une description précise de ces données est fondamentale.

Dans BPMN, les instances des objets de collaboration manipulés par les processus métier sont présents dans le *DataStore*. Les actions du processus permettent de récupérer et de mettre à jour ces instances. BPMN propose aussi le concept *DataObject* (ou donnée du processus), ce qui correspond à nos instances des objets de collaboration manipulés par nos processus. Ces *DataObjects* sont décrits directement dans le processus et représentent les données d'entrées et de sorties des actions métier. Ces concepts ne sont pas assez précis pour permettre de générer le code complet. Par exemple, pour les *DataObject* aucune information sur les propriétés des instances manipulées n'est spécifiée. Pour les *DataObject* produits (c'est-à-dire créés), aucune information sur leurs emplacements dans la structure des données n'est spécifiée.

Dans MACoP, les instances des objets de collaboration manipulés par les processus sont décrites dans un « **DataStore** » sous forme de « **InstanceSpecification** ». Ces instances sont ensuite référencées par des expressions OCL afin de référencer les données concrètement utilisées par les actions métiers. Dans MACoP, une donnée est représentée par le concept « **ElementaryData** » (voir figure 47). Ce concept permet aussi bien de décrire les données primitives, comme un entier ou chaine de caractères, que les instances des objets de collaboration. Par exemple, nous pouvons connecter des « **ElementaryData** » aux paramètres des composants comme expliqué dans la section 6.3.3.

Le concepteur du portail collaboratif, dans MACoP, peut être amené à spécifier des structures de données complexes « **ComplexData** », par exemple des tableaux. Dans MACoP nous introduisons le concept « **InstanceSpecificationFilter** » qui est un sous-type de « **ComplexData** ». Le concept « **InstanceSpecificationFilter** » permet de sélectionner une instance (ou une collection d'instances) d'objet de collaboration, qui est retournée sous la forme de structure de données complexe. Cette structure de données complexe contient l'instance (ou une collection d'instances) ainsi que les attributs associés au type (l'objet de collaboration) de l'instance renvoyée. Ce filtre agit comme une requête sélectionnant les instances qui sont dans l'état demandé par le filtre. La figure 46 illustre la récupération d'un sous

ensemble des attributs d'une facture. Dans cet exemple, nous avons une instance de **Facture facture1** décrivant l'ensemble de ces attributs. Afin de récupérer seulement le **nom** et le **total** de cette facture, nous avons un « **InstanceSpecificationFilter** », **Ma facture** permettant de renvoyer un "tableau" contenant la **facture1** avec ses attribut **nom** et **total**. Ces attributs sont spécifiés dans l'attribut « **attributes** » de « **InstanceSpecificationFilter** ». MACoP ne spécifie pas la nature de la structure de données complexe renvoyée. Sur cette figure, nous avons fait le choix de représenter cette structure de données sous la forme de tableau.

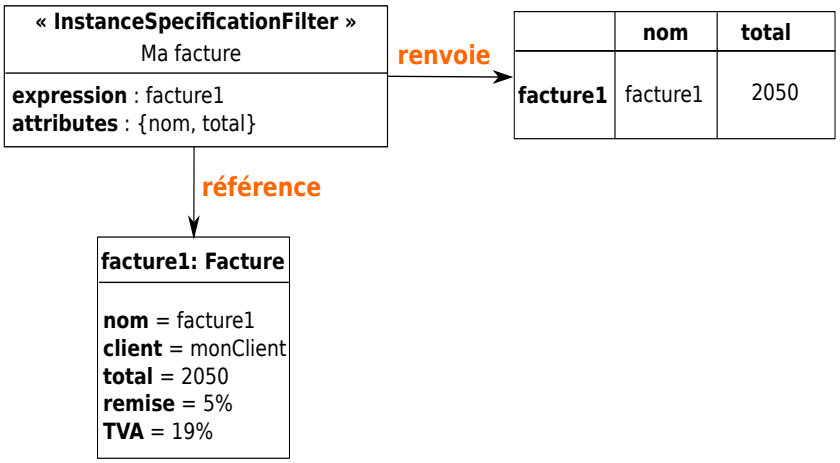


FIGURE 46: La récupération du **nom** et du **total** d'une facture par un *InstanceSpecificationFilter*

Afin de pouvoir utiliser le « **InstanceSpecificationFilter** » comme une donnée, le filtre est un (il est un sous type de) « **Data** » comme illustré à la figure 47.

Dans MACoP, une « **InstanceSpecificationFilter** » peut être utilisée comme une structure de données d'entrée ou de sortie d'une action selon le contexte d'utilisation. Dans le cas où l'« **InstanceSpecificationFilter** » est utilisée comme une structure de données d'entrée, l'utilisateur doit saisir les valeurs des attributs renvoyés par l'« **InstanceSpecificationFilter** ».

Les « **InstanceSpecificationFilter** » sont utilisées afin de décrire les données en relation avec les actions métier. Les relations entre les données et les actions peuvent être définies dans différents contextes comme décrits dans ce qui suit.

Les contextes des actions métier

Lors d'une collaboration, un acteur est souvent amené à étudier un ensemble de données afin de mener à bien son rôle. Dans BPMN une action peut s'appliquer sur un ensemble de données pour en produire un autre. Ni la destination des données produites ni l'ensemble de données à étudier ne sont spécifiés, ce qui ne permet pas de

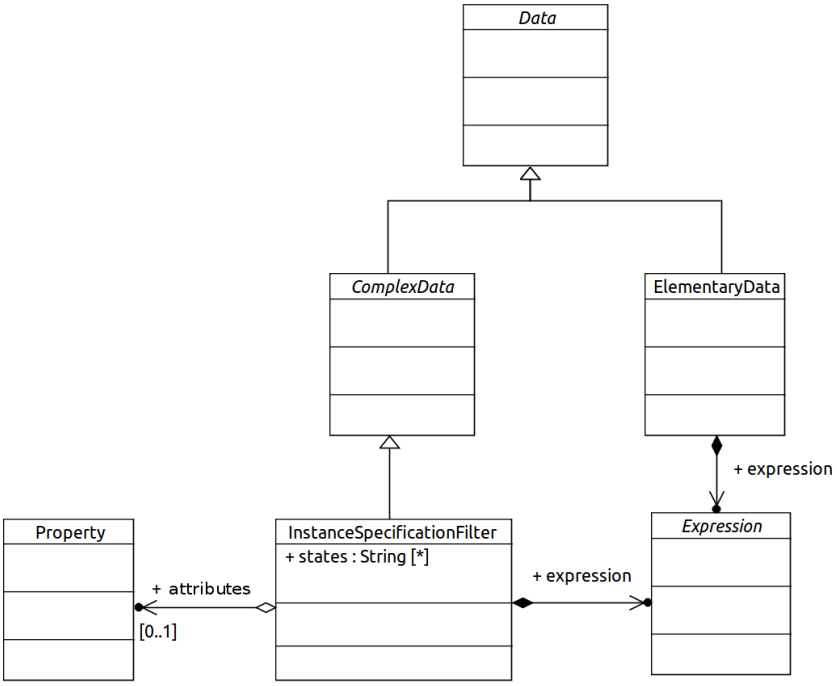


FIGURE 47: Le métamodèle des données

générer du code opérationnel. Nous appelons cet ensemble de spécifications nécessaire au fonctionnement de l'action le contexte d'action. Nous avons donc ajouté la notion de contexte d'action pour les actions métier, ce qui permet de spécifier sur quelle donnée l'action agit et à quelle donnée l'utilisateur du portail collaboratif accède.

Le contexte de l'action *ActionContext* (voir figure 49) est composé de quatre sous-contextes qui sont : le contexte de réalisation *RealizationContext*, deux contextes d'études *StudyContext* et le contexte de production *ProductContext* illustrés à la figure 48.

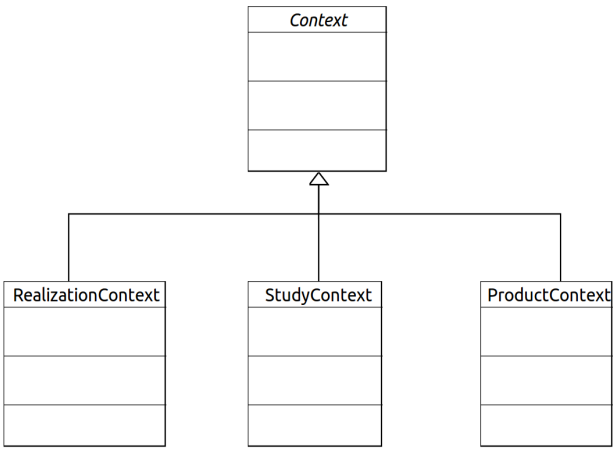


FIGURE 48: Les différents sous-contextes du contexte d'action

– **Le contexte de réalisation :** Dans une collaboration, une action d'un processus métier agit sur plusieurs instances d'objet de col-

laboration. Cet ensemble d'instances est identifié comme le contexte de réalisation. Une action métier se réalise principalement sur une instance d'objet de collaboration bien identifié. Cette instance est identifiée comme le contexte principal du contexte de réalisation. Par exemple, dans un processus de gestion de projet de développement, un projet contient un devis. Un chef de projet doit valider le projet. L'action de validation du projet est déclenchable sur le projet. Cette action agit sur le devis du projet en demandant au chef du projet de vérifier et compléter le devis afin que le projet soit validé. Dans cet exemple, l'action agit sur le projet et sur le devis. Seulement le projet est considéré comme le contexte principal de cette action. De ce fait si le chef de projet étudie le projet, il se voit proposer l'action de validation du projet par l'application. Une fois que l'action est déclenchée, le chef de projet doit modifier le devis.

MACoP dispose du concept « **RealizationContext** » pour décrire le contexte de réalisation. Ce contexte déclare l'ensemble des instances sur lesquelles l'utilisateur peut agir. Ces instances sont déclarées sous la forme de « **InstanceSpecificationFilter** » (voir figure 49). Seule une instance de cet ensemble est considérée comme le contexte principal comme illustré à la figure 49.

Le concepteur peut spécifier, pour chaque « **InstanceSpecificationFilter** », l'état dans lequel l'instance doit être ainsi que les attributs de cette instance qui sont manipulés directement par l'utilisateur dans le cadre de l'action associée au contexte réalisation. Les états des instances définies dans les « **InstanceSpecificationFilter** » sont considérés comme des contraintes qui doivent être évaluées à vrai afin que les utilisateurs puissent déclencher l'action. Dans le contexte de réalisation, les « **InstanceSpecificationFilter** » sont considérés comme des structures des données d'entrées. La spécification de l'état ainsi que les attributs des « **InstanceSpecificationFilter** », sont facultatifs.

Le contexte principal du contexte de réalisation, quant à lui, doit être une des « **InstanceSpecificationFilter** » du contexte de réalisation. Le contexte principal est à ne pas confondre avec les paramètres de l'implémentation de l'action qui définissent les données manipulées par celle-ci. Par exemple, dans un processus de facturation l'action de paiement de la facture peut se réaliser quand l'instance d'objet de collaboration étudiée est la facture manipulée par ce processus. Afin de réaliser l'action, l'utilisateur doit saisir ses coordonnées bancaires. Dans cet exemple le contexte de réalisation spécifie la facture en tant que contexte principal. Les coordonnées bancaires, quant à elles, sont les paramètres d'entrée de l'*implémentation* de l'action (voir section 6.3.3). Le contexte principal, du contexte de réalisation, permet de créer une association entre l'instance d'objet de collaboration, consid-

érée comme le contexte principal, et l'action du processus métier. À partir d'une instance d'objet de collaboration, nous pouvons retrouver toutes les actions qui lui sont associées (dont l'instance est définie en tant que contexte principal). Cela en accédant à la propriété « **actions** » de l'objet de collaboration « **Entity** » présent dans la bibliothèque de MACoP comme décrit dans la section 6.3.4.

- **Le contexte d'étude :** Ce contexte permet de spécifier ce que l'utilisateur de l'application doit étudier avant d'effectuer l'action ainsi qu'après le traitement de l'action, car il doit connaître l'ensemble des données nécessaires à la réalisation de celle-ci. Ces données sont spécifiées à l'aide des « **Data** » (voir figure 47). Le contexte d'action contient deux contextes d'études comme illustré à la figure 49 : le contexte de pré-étude et le contexte de post-étude. Ces deux contextes permettent de spécifier le droit d'accès aux données du portail collaboratif de l'utilisateur concerné. Nous parlons ici de la navigation structurelle contextualisée.
  - *Le contexte de pré-étude* est spécifié par rapport à l'action et permet de décrire la structure de données nécessaire à la prise de décisions. Il est alors possible de déduire des vues spécifiques à l'instance de l'objet de collaboration étudiée avant le traitement de l'action. Par exemple dans un processus d'approvisionnement d'un magasin, le gérant peut effectuer une demande d'approvisionnement à ses fournisseurs. Pour cela, le gérant a besoin de l'état du stock et de la liste des produits épuisés. Ces données constituent le contexte de pré-étude.
  - *Le contexte de post-étude* est spécifié par rapport à l'action et permet de décrire la structure de données résultat du traitement. Il est alors possible de déduire des vues spécifiques à l'instance de l'objet de collaboration étudiée après le traitement de l'action. Par exemple pour l'action de l'exemple précédent, le gérant a besoin des données confirmant sa demande d'approvisionnement. Ces données constituent le contexte de post-étude.
- **Le contexte de production :** Dans une collaboration, un acteur agit sur les instances des objets de collaboration en suivant les actions métier (création, modification, suppression, etc.). Généralement, ces actions produisent d'autres données ou modifient les états des instances des objets de collaboration existantes. La notion d'état permet de simplifier la modélisation et d'exprimer une étape dans le cycle de vie d'une instance donnée. Naturellement, une instance d'un objet de collaboration passe par l'état de « **néant** à un état quelconque (ici la création) puis d'un état quelconque à l'état « **néant** » (ici la destruction). La création d'une instance d'un objet de collaboration est accompagnée par la spécification de l'instance d'objet de collaboration qui va la recevoir,

par exemple **créer** un **fichier** dans un **dossier**. Dans ce cas, le **fichier** passe de l'état « **néant** » à l'état **créé** et sa destination est le **dossier**. Seul l'état « **néant** » est préexistant dans MACoP. Les autres états sont définis librement par le concepteur du portail collaboratif dans son modèle.

Dans MACoP, le contexte de production permet de décrire les états des instances des objets de collaboration, après le traitement de l'action, ainsi que leurs nouvelles destinations en cas de changement de celle-ci. Ainsi, nous avons une idée précise sur les états et l'emplacement des instances des objets de collaboration produits dans l'application.

Dans le cas de notre exemple de création de fichier dans un dossier, le contexte de réalisation permet de spécifier le fichier dans l'état « **néant** », signifiant ainsi qu'il ne doit pas exister. Le contexte de production, quant à lui, permet de spécifier le fichier dans l'état **créé**, à l'aide des « **InstanceSpecificationFilter** », et de définir le dossier comme sa destination, à l'aide d'une expression OCL.

En résumé, dans MACoP les contextes de réalisation et celui de production décrivent comment agir sur les données. Le contexte d'étude quant à lui décrit à quelles données il faut accéder.

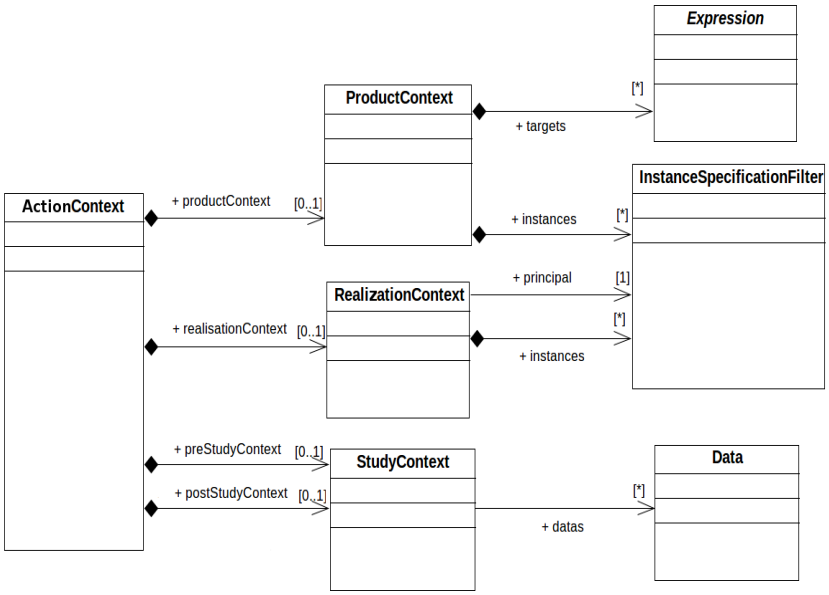


FIGURE 49: Le métamodèle des contextes

Une action peut être associée à plusieurs contextes d'action. Par exemple dans un processus de facturation, l'action de paiement peut être réalisée sur la facture à payer ou sur l'instance de processus lui-même. Dans ce cas, nous avons deux contextes d'action distincts pour l'action de paiement. Le premier contexte d'action contient un contexte de réalisation principal spécifiant la facture, et le deuxième con-

texte d'action contient un contexte de réalisation principal spécifiant le processus lui-même. Ainsi, nous pouvons modéliser un comportement centré sur les données (c'est-à-dire une exécution naturelle des processus métier) pour le premier contexte d'action ou sur le processus (c'est-à-dire une orchestration de processus métier) pour le deuxième contexte d'action.

Les actions métier à déclenchement automatique

Afin de simplifier le travail des collaborateurs de l'entreprise, une solution consiste à automatiser une partie de ses processus métier. Ceci implique l'automatisation des actions des processus métier. Bien sûr, certaines actions ne peuvent être automatisées, par exemple la livraison d'un produit à un client.

Dans BPMN, les actions automatiques sont des actions déclenchées et réalisées par le système (c'est-à-dire la machine sur laquelle le processus métier s'exécute). Ce système n'est pas considéré comme un acteur dans BPMN. Les paramètres d'entrées des actions dans ce cas sont définis au niveau du modèle.

Dans MACoP et pour une action donnée, nous différencions le déclenchement de l'action du déclencheur de l'action. Ainsi, le déclenchement d'une action peut être automatique ou manuel. Le déclenchement automatique signifie que l'action est déclenchée sans l'intervention d'un humain (ici l'humain ne choisit pas de faire l'action). Le déclenchement manuel, quant à lui, nécessite l'intervention de l'humain afin de déclencher l'action (ici l'humain choisit de faire l'action). Le déclencheur de l'action, quant à lui, peut être un acteur humain ou une machine. Le système, dans MACoP, est considéré comme un acteur machine possédant le rôle Système. Une action affectée au rôle Système est alors une action automatique déclenchée automatiquement. Les paramètres d'entrées dans ce cas doivent être spécifiés dans le modèle.

Par contre, une action à déclenchement automatique affectée à un acteur humain est déclenchée automatiquement dès que toutes les contraintes associées à l'action sont satisfaites. Les paramètres d'entrées dans ce cas sont introduits par l'acteur.

En conséquence, le choix du type d'action (action à déclenchement manuel ou à déclenchement automatique), la présence ou non des contextes de l'action (contextes de réalisation, de pré-étude et de post-étude) induit un comportement différent au niveau du rendu de l'application dans les vues. Cette technique nous permet de modéliser des vues associées aux instances des objets de collaboration comme décrit dans la section 6.4.

Pour pouvoir donner à notre métamodèle la possibilité de décrire des comportements complexes, nous nous basons sur la hiérarchisation des processus métier. Deux mécanismes de hiérarchisation sont alors possibles ; les sous-processus et l'imbrication des processus.

La hiérarchisation des processus métier dans MACoP

La hiérarchisation des processus métier consiste à modéliser les processus métier sur plusieurs niveaux de détails. Cette hiérarchisation est utile dans la cartographie des processus métier. La cartographie des processus métier consiste à identifier les processus métier de l'entreprise ainsi que les classifier selon les différents domaines d'activité de l'entreprise. Cette identification va des macroprocessus jusqu'aux processus les plus détaillés. Un macroprocessus permet de décrire l'activité de l'entreprise dans sa globalité. Chaque macroprocessus identifié à un haut niveau d'abstraction est détaillé pour avoir un autre niveau plus détaillé des processus métier. Le niveau de détails de ces processus est à définir par le concepteur des processus métier (l'expert métier).

Dans MACoP, la cartographie des processus métier se fait d'abord par la modélisation des domaines d'activité « Domain ». Ces domaines sont ensuite détaillés en processus domaine « DomainProcess ». Ensuite, chaque processus domaine est détaillé en processus métier (en utilisant le formalisme BPMN 2.0). Ces processus métier eux-mêmes peuvent être hiérarchisés. Cela en utilisant la notion de sous-processus déjà existante dans BPMN et la notion de processus imbriqué. Dans cette section nous commençons par expliquer la notion de sous-processus puis nous détaillons la notion de processus imbriqué que nous avons introduite et enfin nous montrons la différence entre les sous-processus et les processus imbriqués.

La notion de sous processus permet, d'une part, la simplification de la modélisation en cachant une partie du processus derrière une action métier globale (c'est-à-dire à haut niveau d'abstraction) et la hiérarchisation du fonctionnement de la collaboration d'autre part. Dans BPMN, les sous-processus sont déclenchés par une action qui les englobe. Ils sont exécutés par l'utilisateur à qui l'action englobante est affectée. En considérant que l'évolution d'un processus métier se fait dans un contexte d'exécution constitué des différentes données manipulées par celui-ci, un sous-processus hérite automatiquement du contexte d'exécution de son processus parent. Un sous-processus appartient donc à un même processus et permet à celui-ci d'avoir une structure hiérarchique au niveau du fonctionnement.

Dans les métamodèles CIM-OSA et EUML, un processus métier est considéré comme une entité active, et de ce fait il fait partie de la vue fonctionnelle du système d'information. Cependant, un processus métier peut aussi être considéré comme une information passive de l'entreprise, permettant ainsi sa description dans la vue informationnelle du système d'information. En nous basant sur cette constatation, nous proposons qu'un processus métier soit aussi un objet de collaboration (voir « DomainProcess » dans la figure 35). Dans notre approche, l'objet de collaboration « DomainProcess » peut être instancié. Cette instance peut être créée et manipulée par d'autres pro-

cessus métier définis dans le modèle à l'instar des autres instances des objets de collaboration. La création d'un processus métier par un autre, dans notre proposition, est identifiée comme l'imbrication de processus métier. Cette imbrication donne la possibilité de décrire des comportements complexes comme le contrôle et la supervision des processus métier, connue sous le nom de BAM (Business Activity Monitoring), afin de les analyser quantitativement et qualitativement. Cette imbrication nous permet d'avoir une vision plus claire sur le métier de l'utilisateur allant des processus globaux aux processus élémentaires (la cartographie des processus métier). Il devient possible de décrire d'une manière précise la communication générale entre les processus, la communication entre les instances particulières d'un même processus ou entre les instances particulières de processus différents étant donné que les instances des processus sont désignées et accessibles. Afin de contrôler le processus imbriqué, le processus parent doit avoir la possibilité de l'identifier et de le manipuler comme une donnée de l'application. C'est pourquoi on considère ce processus comme un objet de collaboration.

La notion de processus imbriqué ne doit pas être confondue avec la notion de sous-processus. En effet, les processus imbriqués, au contraire des sous-processus, impliquent plusieurs utilisateurs et possède un contexte d'exécution différents de celui du processus parent (c'est-à-dire le processus qu'il l'a déclenché). Le processus enfant (ici le processus imbriqué) est complètement indépendant pour ce qui est des définitions, mais dépendant du processus parent au niveau de la logique métier. Ainsi, un même processus enfant peut être appelé et contrôlé par plusieurs processus parents, ce qui favorise la réutilisation de modèle. Dans notre approche, l'instanciation d'un processus est équivalente à son activation.

Cette séparation entre les différents points de vue (informationnelle et fonctionnelle) de la notion de processus métier est la source de confusion entre le sous- processus et le processus imbriqué. En effet, le sous-processus est souvent confondu avec le processus imbriqué. Par exemple dans la solution proposée par BonitaSoft [18] [45], les sous-processus sont décrits par une action référençant un autre processus. L'activation de l'action provoque celle du sous-processus tandis que la fin de l'action est conditionnée par celle du sous-processus. Le sous-processus dans ce cas est complètement indépendant du processus parent, ce qui correspond au processus imbriqué par rapport à notre analyse alors que pour l'activation et la fin du processus cela correspond à la notion de sous-processus. Le contrôle de ce dernier, dans ce cas, se limite à l'activation.

Dans MACoP, le niveau de détail de la hiérarchisation concerne aussi les actions du processus métier. Ainsi, une action peut être détaillée en lui associant, ce que nous appelons, une *implémentation*. Cette *implémentation* est décrite par un modèle à base de composants.

Implémentation des actions à base de composants

Dans BPMN, une action décrit une tâche à haut niveau d'abstraction dont les détails ne sont pas spécifiés. Généralement, les actions effectuent des calculs complexes propres au métier des acteurs. Par exemple, le calcul d'une remise pour une facture donnée. Les actions peuvent, aussi, faire appel à des services, par exemple la connexion à un agenda externe. Cet ensemble de calculs et de services constitue l'*implémentation* de l'action.

BPMN ne spécifiant pas comment implémenter les actions, nous proposons une approche basée sur des composants. Un composant définit les données requises en entrée, et les données fournies en sortie (c'est-à-dire les paramètres). Ces données peuvent être de n'importe quel type, notamment des objets de collaboration. Le composant est chargé d'appliquer une opération sur les données d'entrée afin de produire les données de sortie. Une *implémentation* d'une action peut alors être réalisée par un ensemble de composants connectés entre eux. Les points de connexion sont les paramètres de ces composants. Une connexion permet de faire transiter les données des paramètres de sortie vers les paramètres d'entrées. Les paramètres d'entrées non connectés doivent être spécifiés par l'utilisateur de l'application. Les paramètres peuvent être connectés aux données de l'application « Data » (voir figure 47), elles-mêmes identifiées et accédées à l'aide d'expressions OCL. En générale, ces « Data » référencent les données de l'application spécifiées dans la partie initialisation de la vue informationnelle, comme vue dans la section 6.3.1, et les données manipulées par le processus.

Comme vue dans la section 6.3.3, les actions ont un contexte d'action permettant de décrire sur quelles données il faut agir à travers le contexte de réalisation et le contexte de production et à quelles données il faut accéder à travers le contexte d'étude. Une action peut avoir plusieurs contextes d'action. Comme illustré à la figure 50, dans MACoP, nous associons l'*implémentation* de l'action au contexte d'action. Les composants de l'*implémentation* d'une action sont des propriétés intellectuelles (IP-Intelectual Properties : l'utilisateur ne connaît que l'interface d'utilisation, l'implémentation interne, qu'il est possible de trouver sur étagère, reste la propriété du concepteur du composant).

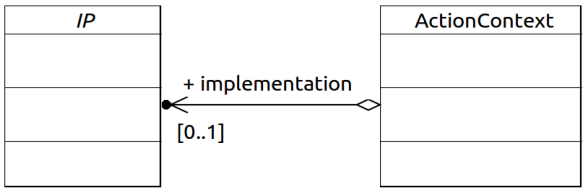


FIGURE 50: L'association *implémentation* entre le contexte d'action et l'IP

L'association des contextes d'action à leurs *implémentations* ainsi que l'assemblage des composants sont fait par l'acteur métier, tandis que l'implémentation du comportement des composants est prise en charge par le concepteur (informaticien) indépendamment de la conception de l'application. Cette implémentation du comportement peut être décrite en interne dans le modèle de l'application (composant interne décrit dans le modèle), à l'aide d'expressions OCL, ou dans une description externe en dehors du modèle de l'application (composant externe trouvé sur étagère).

La description du composant externe est constituée de patrons de génération de code (actuellement en Acceleo [34]) qui peuvent différer en fonction de la technologie ciblée. Ces patrons permettent de prendre en compte les paramètres des composants externes (par exemple le typage) et de générer du code adapté à la spécificité de ces paramètres. Ces paramètres sont des paramètres de configuration du composant externe. Ils doivent être spécifiés dans le modèle du portail collaboratif. Avec ces paramètres de configuration, le composant externe devient générique exprimant de ce fait une intention par exemple l'intention d'**envoyer un message**. En effet, l'envoi d'un message est une intention abstraite dans la mesure ou il existe plusieurs outils et méthodes d'envoi de message (fax, mail, SMS etc.). Ces différents outils et méthodes constituent les types d'utilisation du composant externe. Un composant externe permet de regrouper l'ensemble des différents types d'envoi de message pour laisser le choix au concepteur du portail collaboratif de spécifier le type d'envoi grâce aux paramètres de configuration. Dans notre approche, une transformation de modèle se charge de faire appel aux patrons de génération de code des composants externes, afin de générer leur code, et d'intégrer le code généré au modèle d'entrée du portail collaboratif.

Dans notre approche, un composant externe est représenté par un ensemble constitué d'un modèle décrivant l'interface du composant (les paramètres d'entrées de sorties et de configuration), d'un patron de génération de code décrivant le type d'utilisation désiré selon les paramètres de configuration présents dans le modèle décrivant l'interface du composant et enfin de l'ensemble de fichiers compressés nécessaires au fonctionnement du composant.

Services

Un service permet de regrouper des fonctionnalités réutilisables. Par exemple, un service peut fournir des fonctionnalités d'accès à un agenda externe permettant entre autres l'ajout d'un rendez-vous ou sa suppression. Ces fonctionnalités sont appelables dans les expressions OCL que l'on trouve dans les composants internes.

Un Service est un comportement visible par toute l'application. Il peut être appelé par tout élément du modèle. L'utilisateur modélise la (ou les) fonctionnalité(s) d'un service en utilisant le mécanisme

d'*implémentation* des actions. Contrairement aux *implémentations* des actions, celle des fonctionnalités des services ne contiennent pas de données en relation avec les processus métier étant donné qu'elles sont indépendantes des exigences métier de la collaboration.

6.3.4 *La bibliothèque MACoP*

Généralement, les applications web partagent entre elles des concepts communs. Par exemple le concept d'administrateur de l'application. Ces concepts peuvent être capitalisés afin d'être réutilisés dans de nouvelles applications. Selon nous, ils font partie du modèle décrivant le portail collaboratif et non du métamodèle. Ces concepts constituent les concepts de base d'un portail collaboratif pouvant évoluer dans le temps. Dans MACoP, nous avons identifié les concepts communs entre les collaborations et nous les avons capitalisés dans une bibliothèque réutilisable. Le fait de les capitaliser dans une bibliothèque nous permet de les maintenir plus facilement. Le métamodèle MACoP est, donc, accompagné par une bibliothèque décrivant les concepts abstraits tels que les rôles ou les objets de collaboration de base (voir figure 51) d'un portail collaboratif ainsi que les types primitifs (String, Integer, etc,).

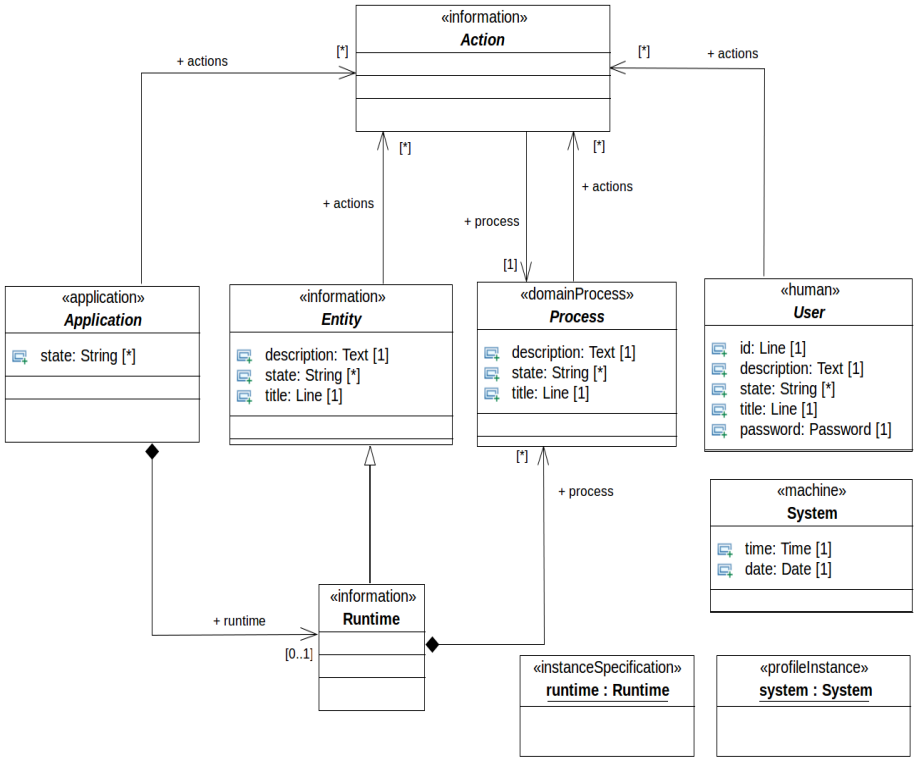


FIGURE 51: Les objets de collaboration abstraits de la bibliothèque MACoP

On y trouve par exemple, le type de profil « **System** » ainsi que son instance « **system** » qui décrit le système avec ses attributs, par

exemple l'heure ou la date. Nous avons affecté le rôle « **System** » (voir figure 52) à l'instance de profil « **system** ». Le profil « **System** » est de type « **Machine** » du métamodèle MACoP. Ainsi toute action affectée au rôle « **System** » est exécutée automatiquement par le système. La bibliothèque dispose aussi d'une application abstraite « **Application** » avec l'attribut « **runtime** », de type « **Runtime** », permettant de stocker les instances des processus définies dans le modèle du portail collaboratif. En effet, dans MACoP, l'initiation d'un processus métier est équivalente à la création d'une instance d'un objet de collaboration de type abstrait « **DomainProcess** ». Si le concepteur ne spécifie pas l'emplacement de ces instances dans la structure documentaire (voir section 6.3.1) alors, l'application se charge de les sauvegarder dans l'emplacement « **runtime** ».

D'autres objets de collaboration abstraits sont disponibles dans la bibliothèque MACoP, comme l'objet de collaboration « **Entity** ». Cet objet de collaboration est une « **Information** » abstraite comportant les différents attributs de base qu'une information dans le modèle du portail collaboratif doit avoir. Par exemple, « **Entity** » définit l'attribut « **actions** » qui décrit l'ensemble des actions à effectuer sur l'information selon les différentes contraintes métier. Nous trouvons aussi, l'attribut « **description** » de type « **String** » permettant d'avoir une description de l'information, l'attribut « **state** » définissant l'état de l'information dans l'application, etc. Toute information dans le modèle du portail collaboratif hérite automatiquement de l'information « **Entity** ». De ce fait, toutes les informations disposent d'un certain nombre d'attributs de base "standardisés".

Dans notre approche, chaque objet de collaboration hérite automatiquement d'un objet de collaboration abstrait correspondant au type de l'objet de collaboration. Ces objets abstraits sont décrits dans la bibliothèque MACoP. On a donc « **Process** » pour les processus, « **User** » pour les profils des utilisateurs humains, « **Application** » pour l'application et « **Entity** » pour les informations.

La figure 52 montre les rôles communs à tous les portails collaboratifs, que nous avons identifiés, décrits dans la bibliothèque MACoP, par exemple « **Administrator** » pour l'administrateur, « **Collaborator** » pour les collaborateurs identifiés d'une manière générale, « **Anonymous** » pour les visiteurs non identifiés. Ces rôles sont utilisés pour la génération des processus par défaut et peuvent être réutilisés par le concepteur de l'application.

Les concepts abstraits et leurs attributs associés peuvent être utilisés dans le modèle du portail collaboratif.

6.4 MODÉLISATION DES ASPECTS VISUELS

Les notions de pages web et de liens de navigation existent dans le portail collaboratif généré. Simplement, elles ne sont pas mod-

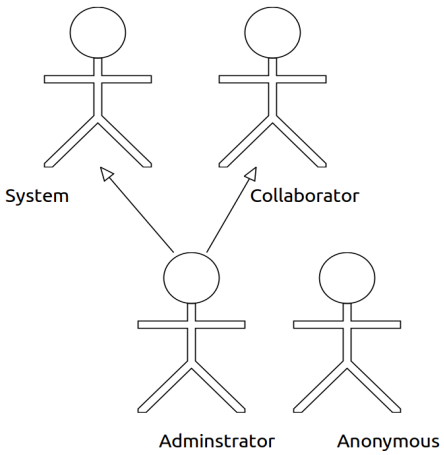


FIGURE 52: Les rôles de la bibliothèque MACoP

élisées explicitement dans des modèles lors de la conception du portail collaboratif. Ces notions sont déduites automatiquement du modèle métier et plus spécifiquement des contextes d'action. Dans cette section nous montrons comment nous passons d'une description métier de la collaboration à l'architecture visuelle de l'application.

Selon nous, dans une application web, les pages web contiennent les données visuelles que l'application montre à l'utilisateur en fonction de divers paramètres, comme les droits d'accès de cet utilisateur. Nous pouvons identifier deux types de données visuelles : les données visuelles d'interaction qui permettent d'agir sur les données de l'application, par exemple les formulaires et les actions ; et les données visuelles informatives qui permettent de donner de l'information à l'utilisateur, par exemple le nom ou le total d'une facture. Le fait de déclencher une action sur une page web (c'est-à-dire cliquer sur un lien de navigation structurelle ou un lien de navigation fonctionnelle), change les données visuelles affichées. Cela peut être vu comme un flux de données visuelles manipulé par les liens structurels et les liens fonctionnels. Un utilisateur navigue sur ce flux selon les différentes contraintes métier. De ce fait il ne peut voir ou agir sur les données de l'application que si cela lui est permis.

Dans notre approche les liens de navigation structurels et les liens de navigation fonctionnels permettent de déclencher des actions des processus métier permettant de renvoyer des données visuelles à l'utilisateur. Selon l'instance de l'objet de collaboration étudiée par l'utilisateur (c'est-à-dire le contexte principal), l'application affiche les données visuelles en fonction des droits de l'utilisateur. Dans notre approche, le déclenchement d'une action, dans l'application, entraine le passage d'un ensemble de données visuelles à un autre. Pour une instance d'objet de collaboration, nous pouvons avoir plusieurs actions à déclenchement automatique entrainant chacune l'affichage d'un ensemble de données visuelles différent. Ces données visuelles forment

les différentes vues du portail collaboratif. Ces vues sont démunies de leurs styles (c'est-à-dire la position dans la page, les couleurs, etc.) qui tiennent selon nous de l'ergonomie du style et non de l'ergonomie métier.

Dans MACoP, à partir des attributs spécifiés dans les « InstanceSpecificationFilter » des contextes de réalisation ainsi que dans les paramètres d'entrées de l'implémentation de l'action, nous déduisons automatiquement le formulaire (données visuelles d'interaction à saisir par l'utilisateur) qui doit être montré lors du déclenchement de l'action. Les contextes d'étude des actions (contexte de pré-étude et contexte de post-étude) quant à eux nous permettent de déduire les données visuelles informatives (permettant de visualiser les données du noyau fonctionnel). Dans MACoP, l'ergonomie du style n'est pas décrite. En effet, MACoP permet la description des aspects métier, mais ne prend pas en charge la disposition des données visuelles ainsi que leurs styles dans les pages du site web généré.

Actuellement, nous générons par défaut le style des différentes structures de données montrées par les actions. Comme décrit dans la section 6.1.5, une page du portail collaboratif généré est composée de trois parties : le tableau de bord qui contient les différentes actions déclenchables sur l'instance d'objet de collaboration étudiée (contexte principal), le navigateur et le corps qui contient les données visuelles montrées par les actions métier (formulaires, contexte de pré-étude, etc.). Pour chaque action des processus métier nous générons une vue pour son contexte de pré-étude et ses paramètres d'entrées (c'est-à-dire le formulaire de l'action) et une autre vue pour son contexte de post-étude. Ces vues sont concaténées et affichées dans le corps de la page et sont associées à l'instance de l'objet de collaboration décrite par le contexte principal de l'action.

La figure 53 montre le processus de réalisation d'une action dans notre approche. Ce processus décrit les différentes étapes suivies par l'utilisateur et l'application afin de réaliser une action. Dans ce processus l'utilisateur du portail collaboratif commence par déclencher l'action, ensuite si l'action demande des paramètres d'entrées ou si l'action a un contexte de pre-étude alors l'application montre à l'utilisateur le formulaire avec le contexte de pre-étude. Dans le cas contraire, l'application exécute l'action directement. Une fois le formulaire et le contexte d'étude affichés, l'utilisateur saisit les valeurs et valide le formulaire, puis l'application exécute l'action. Une fois l'action exécutée, l'application montre le contexte de post-étude à l'utilisateur si l'action en possède un. Sinon le processus de réalisation de l'action est fini. À noter que si l'action est à déclenchement automatique alors l'action « Déclencher l'action » du processus de réalisation est faite automatiquement. Dans ce cas, selon la présence ou pas des contextes d'étude et des paramètres de l'action (issus de contexte de réalisation et de l'*implémentation* de l'action), nous avons une exécution du

processus de réalisation différente. Par exemple, si l'action est à déclenchement automatique et que l'action ne possède qu'un contexte de post-étude alors dès que l'utilisateur étudie le contexte principal de l'action, la vue contenant le contexte de post-étude sera affichée automatiquement. Ce comportement est équivalent à une action de visualisation de l'instance d'objet de collaboration à étudier.

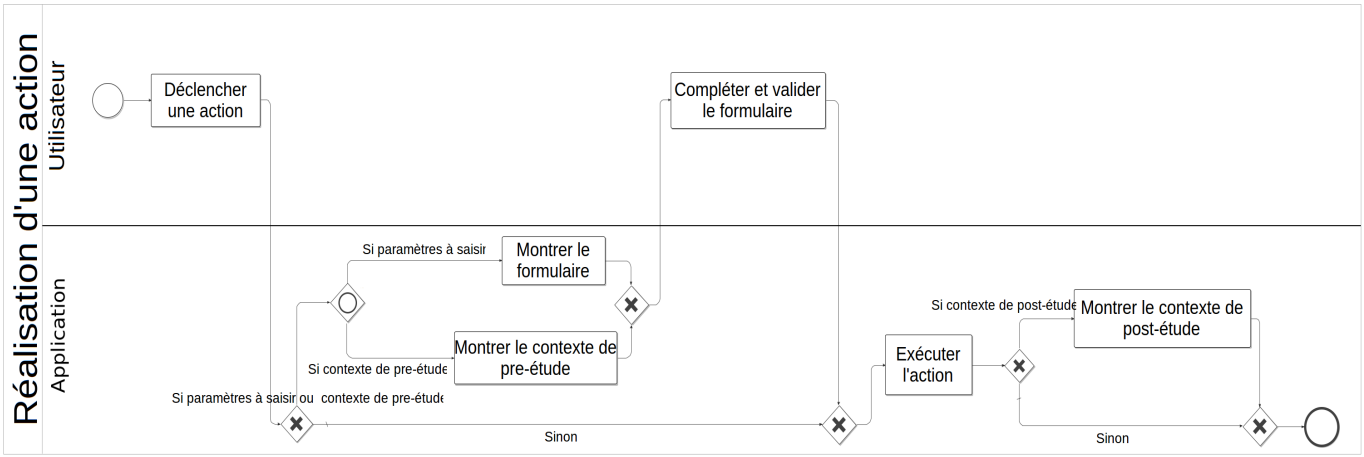


FIGURE 53: Processus de réalisation d'une action

Dans ce qui suit, nous parlons de la modélisation des actions des processus métier nous permettant de déduire les différentes vues du portail collaboratif généré ainsi que la limite de cette thèse concernant l'ergonomie du style.

6.4.1 *Dès actions métier aux données visuelles*

Dans notre approche, le portail collaboratif généré présente les données aux utilisateurs en fonction de leurs rôles, la position dans les processus métier et les différentes contraintes métier. Nous avons vu que la navigation structurelle, permettant à l'utilisateur de se déplacer d'une vue à une autre, est considérée comme une action métier montrant des données visuelles. Ces données visuelles forment les vues en question.

Dans notre modélisation, une action d'un processus métier peut être considérée comme une action de visualisation (une action dont le but principal est de visualiser les données). Ces « actions de visualisation » permettent de décrire quelles données de l'instance d'objet de collaboration étudiée sont montrées, par exemple les valeurs des attributs de l'instance. Elles permettent aussi de montrer des données visuelles différentes de celle de l'instance d'objet de collaboration étudiée afin d'enrichir la vue et de la rendre plus ergonomique au niveau métier. Par exemple, dans une application de réseau social, un utilisateur a un **profil** décrit par un ensemble de données comme son nom et son prénom. Le profil peut contenir une **boîte de récep-**

tion d'**alertes**. Dans notre cas, le profil ainsi que la boîte de réception sont des objets de collaboration. Afin de rendre l'application plus ergonomique, l'utilisateur doit pouvoir voir à la fois les données de son profil et la liste de ses alertes reçues. Sachant que cette liste est sur la boîte de réception, l'application doit renvoyer une vue contenant la liste des alertes en plus de la structure des données du profil.

Une action de visualisation est une action à déclenchement automatique avec un contexte de réalisation spécifiant à quel objet de collaboration elle est associée (le contexte principal) et un contexte de post-étude spécifiant les données devant être montrées à l'utilisateur. Cette action est déclenchée automatiquement dès que l'utilisateur étudie le contexte principal de cette action et que les différentes contraintes de sécurité sont vérifiées. La description d'une telle action nous permet de définir des vues contextualisées. De ce fait, les vues des instances des objets de collaboration deviennent fortement adaptables (selon le rôle, les conditions et la position dans le processus, etc.). Nous parlons ici de l'adaptation des applications web [146]. Une action de visualisation peut-être accompagnée par un traitement comme, par exemple, calculer le nombre des visites pour une instance d'objet de collaboration donnée.

D'une manière générale, nous pouvons associer plusieurs actions à déclenchement automatique au même contexte principal de réalisation. Dans ce cas, toutes les actions dont les contraintes sont évaluées à vrai sont déclenchées en même temps. Au cas où plusieurs actions sont déclenchées en même temps, les structures de données visuelles affichées par ces actions sont concaténées. Par exemple, dans le cas d'une application de gestion de projets de développement, pour un projet donné les développeurs ont accès aux documents d'étude. Le chef développeur quant à lui a accès aux factures en plus des documents d'étude. Dans ce cas, nous avons une action de visualisation ayant les documents d'étude comme contexte de post-étude pour les développeurs et une autre action de visualisation ayant les factures comme contexte de post-étude pour le chef développeur. L'héritage existant entre les rôles, implique l'héritage entre les structures des données de ces actions de visualisation (actions automatiques d'une manière générale), ce qui permet de simplifier la modélisation.

Une action peut avoir comme contexte principal une instance d'objet de collaboration généralisée. Dans ce cas, cette action peut se réaliser sur toutes les instances d'objet de collaboration de même type ainsi que les sous types dès que toutes les autres conditions sont vérifiées. De ce fait, plusieurs vues sur des instances d'objets de collaboration de types différents peuvent partager des structures de données. Ce qui permet de factoriser la modélisation et par conséquent de la simplifier. Par exemple, pour renvoyer un tableau de bord (les actions qu'un utilisateur peut effectuer sur une instance d'objet de collaboration donnée) sur toutes les instances d'objet de collaboration de l'ap-

plication, il suffit de modéliser une action de visualisation avec un contexte principal, une instance « **entity** » généralisée de type « **Entity** » (voir section 6.3.4) et un contexte de post-étude renvoyant la liste « **entity.actions** ».

6.4.2 *Ergonomie du style*

Nous avons constaté que la dynamique de l'IHM peut être complètement déduite du NF ce qui permet de générer une IHM par défaut respectant le métier de l'utilisateur final. Mais, si cette IHM respecte bien le métier final, elle ne répond pas souvent aux critères d'ergonomie attendus par l'utilisateur. Par exemple, la façon dont les données d'entrées sont saisies peut varier selon le type de la plate-forme. Cela constitue le type d'interaction (commande vocale, commande tactile, etc.).

C'est pourquoi, dans MACoP, nous voulons proposer de modéliser l'ergonomie de l'IHM. Ces travaux ne sont pas décrits dans le cadre de ce document. Ils s'appuieront sur les travaux du domaine des IHMs, car d'importantes problématiques sur la présentation et l'interaction entre l'homme et la machine ont déjà été traitées. Ces travaux constituent une approche intéressante pour la modélisation de la partie visuelle des applications web. Ces travaux nous permettrons de nous abstraire du type d'interactions classique, comme les formulaires, et de pouvoir modéliser d'autres types d'interactions plus adaptées à la plate-forme matérielle (commande tactile, vocale, etc.).

Dans la version actuelle de MACoP, nous générons une IHM par défaut. Celle-ci est tout à fait opérationnelle.

6.5 VERS UN MODÈLE D'EXÉCUTION CENTRÉ SUR LES DONNÉES : EXTENSION DU MODÈLE D'EXÉCUTION BPMN

Comme décrit dans la section 6.2.2, dans nos portails collaboratifs, l'exécution des processus métier est centrée sur les données. L'utilisateur, selon son rôle dans la collaboration, effectue des actions sur les instances des objets de collaboration. L'application détermine les actions déclenchables par l'utilisateur en fonction de l'instance d'objet de collaboration couramment étudié et après évaluations des contraintes associées aux actions. Par exemple, lorsque l'utilisateur étudie un devis, l'application peut déterminer que les actions déclenchables sont **accepter** ou **refuser** un devis. Ce type d'exécution est centrée sur les données. Le modèle d'exécution de BPMN ne permettant pas une exécution centrée sur les données, nous devons lui apporter des extensions afin de supporter ce type d'exécution.

Le modèle d'exécution de BPMN est souvent décrit de manière informelle. Dans [147] [109] les auteurs ont essayé de le formaliser. Cette description est fondée sur le concept de Jeton (ce qui correspond *au*

bon de travail comme vue dans la section 2.3). D'une manière générale, un modèle BPMN forme un graphe dont les nœuds représentent des actions ou des branchements reliés par les flux de séquences. Chaque action consomme un ou plusieurs jetons provenant des flux entrants pour produire un jeton sur chaque flux sortant. Le rôle des nœuds de branchement est d'orienter, autoriser ou refuser automatiquement le passage des jetons. On trouve, par exemple, le nœud de branchement parallèle qui permet d'attendre tous les jetons issus des flux entrants pour produire un jeton sur chaque flux sortant. On trouve aussi le nœud de branchement exclusif qui permet de consommer le premier jeton entrant pour produire un seul jeton sur le premier flux qui vérifie la contrainte (c'est-à-dire *guard* sur le flux de séquence). À l'exécution du processus, le nœud de branchement exclusif permet au système de choisir automatiquement l'un des flux sortants permettant d'exécuter la prochaine action. L'utilisateur, dans ce cas, ne peut pas lui-même choisir l'action à exécuter. En effet, dans le modèle d'exécution du métamodèle BPMN et pour les nœuds de branchement exclusifs, le choix du chemin parcouru par les jetons est fait d'une manière automatique selon les contraintes des flux de séquence. De ce fait, l'utilisateur ne peut pas décider de manière explicite, à l'exécution, du chemin parcouru par les jetons. Dans ce modèle d'exécution, les choix de l'utilisateur sont les résultats des actions et sont décrits sous forme de variables. Une contrainte sur un flux de séquence porte sur ces variables et permet de bloquer ou d'autoriser le passage de jeton selon son évaluation. Par exemple dans un processus de gestion de voyages, un utilisateur est amené à faire le choix entre voyager en train ou voyager en avion. Dans le modèle d'exécution BPMN qui est centré sur les tâches, le choix entre les deux types de voyage est fait à travers une action du processus en fixant une valeur dans le système. Selon cette valeur un nœud de branchement exclusif permet d'orienter l'exécution du processus.

Afin d'avoir une exécution centrée sur les données, nous pensons qu'un nœud de branchement exclusif à choix manuel permettant d'orienter les flux d'exécution du processus manuellement est nécessaire. Dans notre contexte, manuellement signifie que le choix du flux de séquence sur lequel le jeton doit passer est fait, de manière interactive, par l'utilisateur de l'application, ce qui n'est pas le cas dans le modèle d'exécution BPMN où le choix du flux de séquence est fait automatiquement par le moteur d'exécution du processus.

Cette section décrit les extensions que nous avons apportées au moteur d'exécution BPMN afin de supporter la possibilité d'avoir des choix exclusifs manuels à l'exécution du processus. Ainsi, nous avons explicité les choix en les externalisant des actions.

6.5.1 *Externalisation des choix*

Dans notre modèle d'exécution centrée sur les données, les actions sont associées aux instances des objets de collaboration (voir le contexte principal dans la section 6.3.3). Ainsi, le choix exclusif manuel de faire une action ou une autre sur les instances (c'est-à-dire décider du flux de séquence à suivre par le jeton) doit être possible. C'est à l'utilisateur, dans ce cas, de valider ou non un flux de séquence (c'est-à-dire le passage d'un jeton).

Un nœud de branchement exclusif doit permettre de valider le choix manuel de l'utilisateur. Les contraintes, dans ce cas, sont implicites et portent sur la validation d'un des flux de séquences sortants par l'utilisateur lors de l'exécution du processus. Par exemple, la figure 54 (a) montre une partie du modèle BPMN de gestion de voyage dans une entreprise. Le collaborateur commence par soumettre une demande, ensuite l'assistant doit valider la demande en choisissant entre « Refuser » ou « Accepter ». Dans BPMN, ce choix doit être le résultat d'une action « Décision ». Ensuite selon ce choix, le nœud de branchement oriente automatiquement la circulation du jeton.

L'externalisation des choix consiste à représenter chaque choix par une action. Le choix, dans ce cas n'est plus une variable manipulée par le moteur d'exécution afin d'orienter la circulation des jetons, mais une action qu'un utilisateur peut choisir de faire ou non. Le chemin parcouru par le jeton résulte alors du choix fait par l'utilisateur. La figure 54 (b) décrit une externalisation des choix équivalente à la figure 54 (a). Cette solution rend l'exécution du processus plus intuitive pour les acteurs métier et plus centrée sur les données.

Dans notre modèle d'exécution, une fois le jeton arrivé au nœud de branchement exclusif à choix manuel, le moteur bloque ce jeton sur le nœud de branchement et produit, ce que nous appelons, un jeton de décision sur chaque flux sortant dont la contrainte est évaluée à vrai. Ces jetons de décision arrivent, ensuite, sur les actions ce qui permet à l'application de proposer ces actions à l'utilisateur. Quand celui-ci a choisi et exécuté une action, le jeton bloqué sur le nœud de branchement passe par le flux correspondant à l'action choisie et tous les jetons de décision sont supprimés.

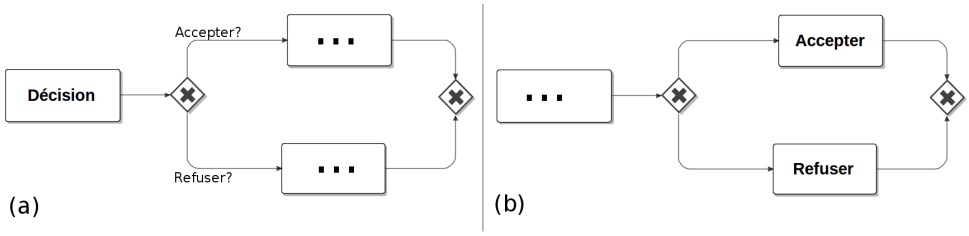


FIGURE 54: Exemple d'externalisation des choix

6.5.2 *Synchronisation de l'évaluation des contraintes*

Avec l'externalisation des choix, le modèle d'exécution de BPMN est devenu insuffisant. Afin de comprendre cela, nous illustrons l'exemple de la figure 55. Dans cet exemple, l'utilisateur a la possibilité de détruire le conteneur s'il est vide ou de transférer son contenu. La gestion du contenu du conteneur est décrite dans d'autres processus, ainsi nous avons une concurrence sur une même ressource (le conteneur).

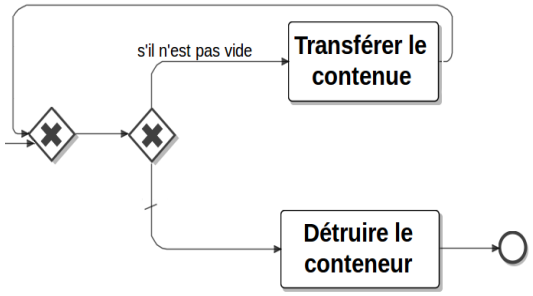


FIGURE 55: Exemple de choix synchrone

Le contenu dans cet exemple évolu constamment. Dans BPMN, les contraintes sur les flux de séquence sont évaluées de manière asynchrone (c'est-à-dire que l'évaluation d'une contrainte ne se fait pas à chaque évolution des données de l'application dans le temps). Une fois l'action précédente terminée, le jeton produit arrive au nœud de branchement exclusif. Celui-ci produit un jeton sur le flux qui vérifie la contrainte. L'action qui reçoit un jeton peut alors être réalisée. Le changement de l'état du conteneur par les autres processus ne change pas la position du jeton, ce qui peut provoquer des erreurs au niveau métier. Par exemple, en supposant que le conteneur est vide, le processus va proposer de supprimer le conteneur. Si un autre processus ajoute un contenu au conteneur, le jeton ne change pas de position et nous aurons toujours le droit de le supprimer parce que la contrainte n'est pas réévaluée. De ce fait, l'utilisateur pourrait supprimer un conteneur qui n'est pas vide ce qui constitue une erreur. Nous avons, donc, besoin d'évaluer certaines contraintes des flux de séquence sortants d'un nœud de branchement exclusif à choix manuel à chaque fois que l'utilisateur est confronté à ce choix (c'est-à-dire les différentes actions proposées par le nœud de branchement exclusif à choix manuel). L'ensemble des actions proposées par ce nœud change selon l'évaluation des contraintes de ses flux sortants. Dans notre exemple, à chaque fois que l'utilisateur est confronté au choix entre détruire le conteneur et transférer le contenu la contrainte vérifiant si le conteneur n'est pas vide doit être réévaluée par le moteur. Dans ce cas, l'utilisateur ne peut pas détruire un conteneur non vide. L'évaluation des contraintes, dans ce cas, doit être toujours synchronisée par rap-

port aux changements des données de l'application. Pour cela nous identifions ce type d'évaluation comme l'évaluation synchrone. Dans notre approche, nous avons étendu le modèle d'exécution BPMN en ajoutant ce type d'évaluation. L'évaluation asynchrone est toujours possible avec notre approche.

Afin de permettre à notre moteur de faire la différence entre une évaluation asynchrone et une évaluation synchrone des contraintes des flux de séquences, nous avons ajouté une métadonnée sur le concept de flux de séquences permettant de préciser si l'évaluation de la contrainte est synchrone ou non. L'ajout d'une condition sur les actions ne permet pas de résoudre le problème. En effet, cela permettra de valider ou non l'action, mais le jeton de décision quant à lui reste bloqué sur l'action. De ce fait, les autres actions ne pourront plus bénéficier d'un jeton de décision.

Comme vue précédemment, dans notre modèle d'exécution, le nœud de branchement exclusif à choix manuel permet au moteur de produire des jetons de décision sur chaque flux sortant dont la contrainte est évaluée à vrai. Ces jetons de décision arrivent ensuite sur les actions. Une fois, les jetons de décision arrivés, ils ne changent pas de position en attendant la décision de l'utilisateur de choisir une action parmi celles proposées. Avec les évaluations synchrones des contraintes des flux de séquences, les jetons de décision ne sont pas bloqués sur les actions, mais sont recalculés par le moteur à chaque exécution (c'est-à-dire à chaque fois que l'utilisateur est confronté aux choix du nœud de branchement exclusif à choix manuel).

6.6 MACOP ET LA RÉUTILISATION DE MODÈLES

Les entreprises réutilisent souvent des processus métier conventionnels et standardisés comme les processus de facturation des clients ou les processus associés au soutien logistique. Cela permet à l'entreprise d'aligner son fonctionnement sur des standards existants et de les adapter à ses besoins. Nous parlons ici de la réutilisation du métier existant. Cette réutilisation permet de mettre en place, en un temps réduit, une stratégie de modélisation par assemblage de composants métier. Afin de bénéficier de ces avantages, notre approche doit permettre la réutilisation et la composition de modèles métier existants.

On peut considérer deux types de réutilisations : la réutilisation interne et la réutilisation externe. Dans la réutilisation interne, ce sont les concepts du même modèle qui sont réutilisés. L'héritage dans UML est une forme de réutilisation interne. Dans la réutilisation externe, des concepts définis dans d'autres modèles sont réutilisés. Ceci se fait essentiellement par la composition de modèles. Ce dernier type de réutilisation est un domaine de recherche récent dans l'IDM et vise à combiner deux modèles ou plus afin d'en constituer un seul [83]. À

noter que la réutilisation est un concept plus abstrait que la composition. En effet, la composition est une forme de réutilisation, l'héritage multiple en est un exemple. Par contre l'héritage simple (c'est-à-dire l'héritage d'un seul concept) est tout simplement de la réutilisation.

On peut distinguer différentes classes de composition de modèles [137]. Nous nous sommes intéressés à la composition par transformation qui transforme la structure des modèles composés, par exemple la fusion ou le tissage de modèles, et celle qui préconise l'établissement des relations entre les modèles, appelée composition pure.

6.6.1 *Composition par transformation*

Dans cette classe nous nous sommes intéressés à la modélisation orientée aspect (MOA) qui est une forme de composition générique définie à un niveau déclaratif et qui s'applique sur un *pattern*. La modélisation orientée aspect est un concept plus abstrait que la programmation orientée aspect (POA) [97]. Ce paradigme a comme finalité la séparation des préoccupations. Pour la programmation cela peut avoir une utilité significative par contre pour la modélisation, où la séparation des préoccupations est considérée comme une motivation principale, ce paradigme converge et se confond à la simple composition de modèles. Pour la modélisation orientée aspect, cette composition est faite automatiquement. Dans ce qui suit, nous détaillons la notion de modélisation orientée aspect basée sur les transformations de modèle. Enfin, nous parlons de ce type de modélisation dans MACoP.

Modélisation orientée aspect

La modélisation orientée aspect requiert, généralement, un modèle d'entrée (c'est-à-dire le modèle de base), le modèle de l'aspect à appliquer (c'est-à-dire le greffon), les points d'insertion qui permettent de sélectionner les éléments du modèle de base (appelés points de jonction) à remplacer ou étendre selon le modèle du greffon. Dans [65] l'auteur ajoute la notion de règle de correspondance permettant d'identifier avec précision les points d'insertion dans le modèle de base et le modèle du greffon. Cela afin d'éviter les problèmes d'identification des points de jonction après application successive de plusieurs aspects ayant les mêmes points d'insertion. Enfin le tisseur qui permet d'appliquer l'aspect sur le modèle d'entrée selon les points d'insertion. Cette application se fait par composition de modèles. Dans ce cas, nous avons une composition de modèles entre le modèle de base et le modèle de l'aspect suivant des règles de transformation décrites par les points d'insertion et les règles de correspondance. La relation entre les différents modèles impliqués dans cette composition peut avoir plusieurs formes allant des relations de référencement à la simple fusion [137].

La modélisation orientée aspect ne doit pas être confondu avec la modélisation de la POA comme décrite ici [153]. En effet, la modélisation de la POA permet de remonter le concept issu de la POA au niveau du modèle UML afin de concevoir des applications orientées aspect en utilisant des technologies comme AspectJ [10]. Plusieurs approches de modélisation des applications web ont intégré la modélisation orientée aspect. Par exemple, dans [14] les auteurs proposent une extension pour UWE [76] permettant d'intégrer la modélisation orientée aspect pour le modèle de navigation. Un autre exemple est AspectWebML [124] qui est une extension apportée au métamodèle WebML [20] afin d'intégrer la modélisation orientée aspect. Dans AspectWebML, les aspects peuvent s'appliquer sur tous les concepts du métamodèle WebML et non seulement sur le modèle de navigation comme c'est le cas pour l'extension apportée à UWE. Plusieurs autres travaux [65, 151, 75] ont porté sur la modélisation orientée aspect. Ces travaux sont adaptés au langage de modélisation UML et restent spécifiques à des diagrammes bien définis comme le diagramme de séquence ou le diagramme de classe.

La technique de composition entre le modèle de base et le modèle de l'aspect est fondée sur la transformation de modèle. Nous pouvons identifier deux types de transformations utilisées pour cette technique : les transformations concrètes (c'est-à-dire décrite à un bas niveau d'abstraction, par exemple une transformation QVT) et les transformations abstraites (c'est-à-dire décrite à un haut niveau d'abstraction).

Si la transformation utilisée par la technique de modélisation orientée aspect est concrète alors le modèle de l'aspect, les points d'insertion ainsi que les règles de correspondance sont tous définis dans la transformation elle-même. La transformation dans ce cas, prend le modèle de base en entrée et le transforme en lui ajoutant les concepts du modèle de l'aspect. Les points d'insertion sont identifiés automatiquement par la transformation à partir du modèle de base. Par exemple, cette solution a été explorée dans les travaux faits dans [89]. Ces travaux permettent d'appliquer l'aspect sécurité sur les unités de calcul électroniques. Cet aspect est appliqué par transformation concrète de modèle.

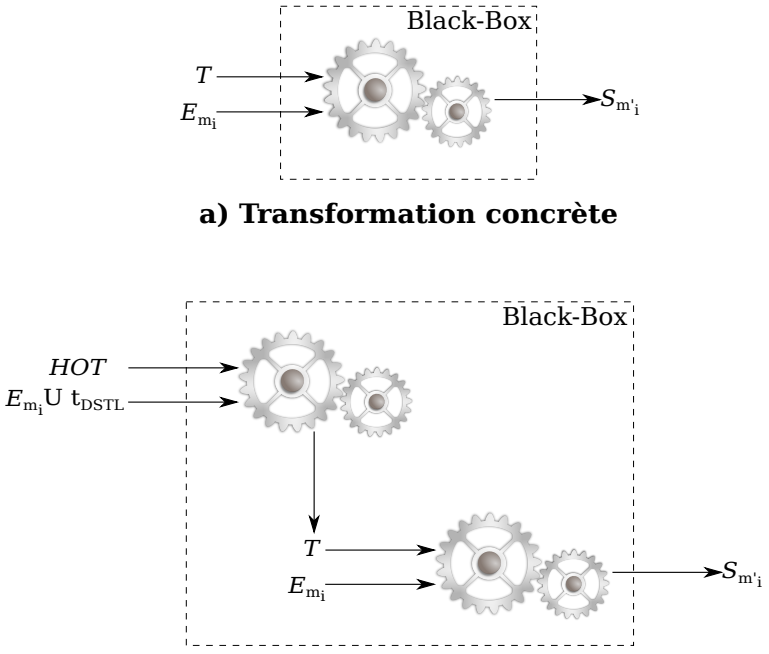
Si le modèle de l'aspect, les points d'insertion ainsi que les règles de correspondance sont définis dans le modèle de base ou dans des modèles indépendants à un haut niveau d'abstraction alors le modèle de l'aspect, les points d'insertion ainsi que les règles de correspondance forment ce que nous appelons une transformation abstraite spécifique au domaine (c'est-à-dire que la transformation est écrite dans un langage de transformation spécifique au domaine (DSTL)) permettant de décrire la composition de modèles entre le modèle de l'aspect et le modèle de base à un haut niveau d'abstraction. Ce type de transformation est utilisé, par exemple, dans la composition de

modèles comme décrit ici [41] ou le *versionning* des modèles comme décrit ici [82].

Généralement, un DSTL ne possède pas un moteur de transformation. Pour combler ce manque, une solution consiste à écrire une transformation **HOT** (Higher-Order Transformation) permettant de transformer une transformation DSTL en une transformation concrète de modèle, décrit par les métamodèles QVT ou ATL par exemple, afin de tirer profit de leurs moteurs d'exécution. La transformation **HOT** est souvent écrite une seule fois. Dans la figure 56, nous montrons la relation entre une transformation abstraite t_{DSTL}, la transformation HOT et une transformation à bas niveau d'abstraction (ou transformation concrète) T. Dans la figure 56 **(a)** nous avons une exécution classique permettant d'appliquer les règles de transformation, de la transformation T, sur l'ensemble de modèles d'entrée E_{m_i} afin de produire un ensemble de modèles de sortie S_{m_i}. Dans la figure 56 **(b)**, nous décrivons comment une transformation abstraite est exécutée. Sur cette figure les règles de transformation de la transformation T sont écrites dans un DSTL formant ainsi la transformation abstraite t_{DSTL}. La transformation HOT dans ce cas, prend la transformation t_{DSTL} comme modèle d'entrée avec l'ensemble de modèles d'entrée E_{m_i} de la transformation T. À partir de la transformation t_{DSTL}, la transformation HOT produit la transformation T. Celle-ci est ensuite appliquée sur l'ensemble de modèles d'entrée E_{m_i} pour produire les modèles de sortie S_{m_i}. Nous avons, donc, un enchainement de deux exécutions ; la première permettant de transformer t_{DSTL} en T et la deuxième permettant de transformer les E_{m_i} en S_{m_i}.

La modélisation orientée aspect dans MACoP

La modélisation orientée aspect dans MACoP a comme finalité la réutilisation ainsi que la composition de modèles (ou aspect) déjà existants (c'est-à-dire sur étagère). Le but est de permettre au concepteur de modéliser leur portail collaboratif par simple assemblage de modèle. Afin de permettre cet assemblage dans notre approche, un aspect peut être appliqué sur un point d'insertion d'un autre aspect. Cela permet d'appliquer des aspects sur des concepts présents dans d'autres aspects. Dans MACoP l'application des aspects est faite selon un ordre bien défini. Cet ordre est défini par les relations entre les aspects. Une relation entre aspects dans MACoP est soit une relation d'ordre, permettant d'indiquer l'ordre d'application des concepts, ou une correspondance entre les points d'insertion appartenant à des aspects différents. Par exemple, pour manipuler une information de la collaboration, on veut avoir un processus permettant de la gérer (aspect gestion de l'information) et on veut avoir un comportement pour la gestion de l'historique de cette information (aspect gestion de l'historique). Dans ce cas il faut appliquer l'aspect gestion



T : Transformation concrète
E_{m_i} : Ensemble des modèles d'entrée
$S_{m'_i}$: Ensemble des modèles de sortie
t_{DSTL}: Transformation abstraite

FIGURE 56: DSTL

de l'information avant l'aspect gestion de l'historique. En effet, l'application de l'aspect gestion de l'historique va modifier les actions qui changent les paramètres de l'information. Si le processus de gestion n'existe pas alors l'application du deuxième aspect ne servira à rien. Un autre exemple consiste à ajouter l'aspect gestion des réunions ainsi que l'aspect gestion de l'historique appliqué aux réunions. Le premier aspect (celui de gestion des réunions) peut déclarer un point d'insertion **P1** qui est un objet de collaboration qui sera identifié comme étant l'objet de collaboration Réunion présent dans le modèle de base. Dans le cas où aucun objet n'est en correspondance avec ce point d'insertion alors l'objet de collaboration Réunion sera ajouté automatiquement au modèle de base. L'aspect gestion de l'historique, quant à lui, déclare un point d'insertion P2 qui est un objet de collaboration qui sera identifié comme étant l'objet de collaboration sur lequel la gestion de l'historique sera appliquée. Le fait de décrire une correspondance de **P1** vers **P2** permet de spécifier que l'aspect gestion de l'historique s'applique sur l'objet de collaboration Réunion de l'aspect gestion des réunions.

La figure 57 montre le métamodèle des aspects dans MACoP. Un aspect « **Aspect** » est modélisé comme une boîte noire (c'est-à-dire que le concepteur du portail collaboratif ne voit pas l'intérieur de l'aspect)

contenant les points d'insertion « **Pointcut** » où l'aspect doit être appliqué. Le concepteur peut spécifier l'ordre d'application en reliant les différents aspects entre eux afin de former un graphe orienté non cyclique (c'est-à-dire un processus d'application des aspects). Cela en spécifiant le flux de contrôle « **ControlFlow** » d'application des différents aspects dans le modèle. Ensuite, le concepteur doit spécifier les règles de correspondance « **Sending** » entre les concepts du modèle de base et les points d'insertion des aspects. Une correspondance peut être spécifiée entre les points d'insertion des différents aspects. Cela permet de décrire toute une application par un simple assemblage de modèles (c'est-à-dire la composition de modèles).

Techniquement, un aspect est généralement formé de trois modèles :

- Le modèle décrivant les métadonnées associées à l'aspect par exemple la version, l'identifiant, les descriptions des points d'insertion. Ce modèle nous permet de référencer et réutiliser l'aspect dans le modèle de base.
- Le modèle qui décrit le greffon. Celui-ci peut être décrit dans la transformation de tissage sous forme de règles de transformation (le modèle du greffon est, dans ce cas, construit par la transformation de tissage).
- Une transformation de tissage permettant d'enrichir le modèle de base avec les concepts décrits dans le modèle du greffon selon les règles de correspondances et les règles de tissage.

Notre outil propose deux modes d'application d'aspect :

- L'application directe (appelée, aussi, synchrone) décrite par [89], dans ce cas l'outil de modélisation se charge de proposer les différents aspects à appliquer sous forme d'actions. Une action permet de déclencher la transformation de tissage sur le modèle de base. Les concepts de l'aspect sont ainsi ajoutés au modèle de base. Ce mode d'application permet au concepteur d'adapter les aspects à ses besoins au niveau du modèle. Ce qui rend la gestion des aspects (ajout, suppression, ordonnancement, etc.) difficile.
- l'application indirecte (ou asynchrone), dans ce cas les différentes transformations de tissage seront appelées au cours du processus de production de logiciels. Ce mode d'application permet une gestion simple des aspects.

6.6.2 *Composition pure*

La composition pure est une composition qui, généralement, ne change pas les concepts du modèle, mais les complète ce qui n'est pas le cas pour la composition par transformation où les concepts du modèle peuvent changer. La composition pure se base sur des techniques basiques comme le référencement. Un des concepts les plus célèbres dans cette catégorie est le composant. En effet, la modélisa-

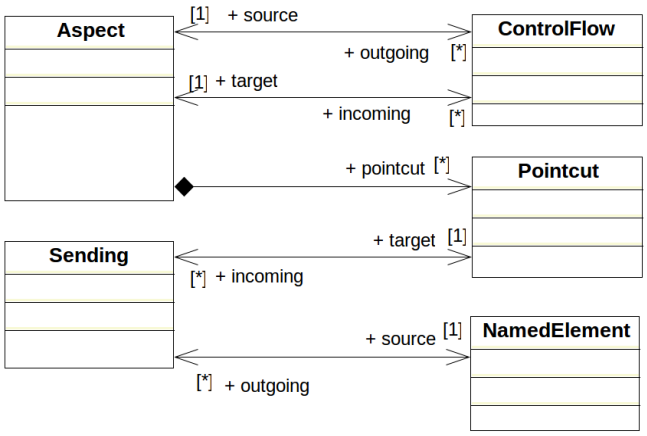


FIGURE 57: Métamodèle des aspects

tion à base de composants permet de construire un modèle par assemblage des composants en les connectant les uns aux autres. Comme décrit dans la section 6.3, nous avons adopté une modélisation à base de composants pour la description des comportements spécifiques des actions métier et des fonctionnalités des services.

6.7 CONCLUSION

Dans ce chapitre, nous avons présenté le métamodèle MACoP qui nous permet de décrire les portails collaboratifs au niveau des exigences métier. Cette modélisation est basée sur la modélisation des systèmes d'information, centrée sur les processus métier. Dans MACoP la modélisation du noyau fonctionnel est divisée en trois vues décrites dans la norme ENV 40003. Ces trois vues sont la vue informationnelle pour la description du plan documentaire de l'application, la vue des ressources pour la description des rôles et enfin la vue fonctionnelle pour la description des processus métier et leur hiérarchisation. En effet, pour les processus métier, nous avons adopté une décomposition hiérarchique allant des domaines d'activité jusqu'à l'implémentation à base de composants en passant par les processus métier décrits par le métamodèle **BPMN2.0**. Dans notre approche, la description de ces processus métier est un mélange entre l'approche orientée tâches et l'approche orientée données (c'est-à-dire orientée artefacts métier). Cela nous donne une idée précise de la vue globale du métier de l'utilisateur, mais aussi du cycle de vie des objets de collaboration. Afin d'ajouter l'aspect centré sur les données au métamodèle BPMN, nous avons dû l'étendre avec des concepts métier, par exemple le choix exclusif manuel.

Dans notre approche, nous ne modélisons pas les pages web et les liens de navigation comme c'est le cas des approches classiques de modélisation des applications web. Dans MACoP les portails collabo-

ratifs sont vus comme des graphes dont les nœuds sont les instances des objets de collaboration et les arcs sont les actions déclenchables à partir de ces instances. Sur chaque instance nous pouvons avoir des vues présentant les données de l'application selon le rôle de l'utilisateur connecté ainsi que selon les différentes contraintes métier. De ce fait, le portail collaboratif est vu comme un flux de données visuelles, contrôlé par les actions des processus métier. Afin de modéliser ce flux dans MACoP, nous nous basons sur les contextes d'action nous permettant de spécifier sur quelles données l'utilisateur doit agir et à quelles données il peut accéder. Dans notre approche, seule l'ergonomie métier est prise en compte, l'ergonomie du style n'est pas traitée par MACoP (ou "pas modélisées dans MACoP") et constitue une perspective de ce travail.

Afin d'augmenter l'expressivité de notre métamodèle MACoP, nous avons utilisé des techniques de modélisation comme l'imbrication de processus (processus imbriqué) ou l'utilisation du langage OCL pour la description des données et des contraintes métier. Le langage OCL est utilisé pour ajouter une touche de précision sur les besoins métier à modéliser et est considéré comme une formalisation des besoins de l'utilisateur.

Afin de simplifier la modélisation et de réutiliser des modèles existants, nous avons proposé une approche de modélisation orientée aspect nous permettant ainsi de réduire la quantité de travail du concepteur.

Dans le chapitre suivant nous décrivons l'outil de modélisation et d'exécution des processus de production de logiciels que nous avons conçu et développé (voir section 3.3) ainsi que notre processus de production nous permettant générer des portails collaboratifs opérationnels en Python à partir d'un modèle MACoP.

7

PMS+ : UN OUTIL POUR ALLER DES MODÈLES AU CODE DE L'APPLICATION

Sommaire

Dans le chapitre précédent, nous avons décrit notre vision de la modélisation des portails collaboratifs. Cette modélisation, à travers le métamodèle MACoP, adopte le langage des entreprises est est centrée sue les processus métier. Afin de produire nos portails collaboratifs, nous adoptons une approche IDM. À partir d'un modèle décrit par le métamodèle MACoP nous produisons le portail collaboratif correspondant.

Comme décrit dans la section 3.1, le passage du modèle à l'application peut être fait soit par compilation (c'est-à-dire par transformation de modèles) soit par interprétation du modèle d'entrée. Dans notre approche nous nous sommes intéressés à la compilation de modèle pour les avantages qu'elle offre [64], notamment la possibilité d'optimiser le code généré. La compilation est généralement concrétisée par un processus de production de logiciels permettant d'enchainer les différentes activités à appliquer sur le modèle d'entrée afin de produire le code source de l'application. Selon nous, l'enchaînement de ces activités (c'est-à-dire des transformations de modèles, des générations de code, etc.) doit être conditionnel afin d'orienter le flux d'exécution des activités. Les conditions dans ce cas peuvent porter sur

les modèles manipulés. De ce fait, poser des questions au niveau du processus de production doit être possible. Les activités quant à elles doivent être réutilisables et hétérogènes (c'est-à-dire utilisant des technologies différentes). Le métamodèle permettant de décrire les processus de production de logiciels dans ce cas doit être indépendant des technologies utilisées par les activités. De ce fait, il faut séparer la technologie du métamodèle. Enfin, un processus de production de logiciels doit permettre une gestion simple des interactions tant avec l'utilisateur qu'avec le système de fichier.

Nous avons vu dans la section 3.3 que les approches actuelles d'automatisation de ce type de processus ne couvraient pas tous nos besoins. Par exemple, la plupart de ces approches sont dépendantes des technologies des activités, par exemple le moteur Acceleo [34] pour l'enchainement des patrons de génération de code, l'outil ATLFlow [113] pour les transformations ATL ou UniTI [145] dont le métamodèle dépend des technologies utilisées par les activités. C'est pourquoi nous avons développé notre propre moteur d'exécution de processus, couvrant nos besoins, et même un peu plus.

Dans ce qui suit, nous présentons notre outil **PMS+** (Process Manufacturing Software and +) qui permet de décrire les processus de production de logiciel 3.3 vérifiant les points décrits dans la section 3.4, à l'aide du métamodèle MPMS (Modeling of Process Manufacturing Software), ainsi que leur exécution. Nous commençons dans la section 7.1 par un exemple illustrant le principe général du moteur. Puis nous décrivons dans la section 7.2 les principaux concepts de notre outil. Dans la section 7.3 nous expliquons le métamodèle MPMS utilisé par le moteur. Suit la section 7.4 une présentation des points techniques particuliers du moteur. Enfin, dans la section 7.5, nous détaillons le processus de production réalisé avec **PMS+** et nous permettant de passer d'un modèle MACoP à un portal collaboratif dont le code Python a été automatiquement généré.

7.1 PRINCIPE GÉNÉRAL DU FONCTIONNEMENT DU MOTEUR

Nous allons illustrer le fonctionnement de notre moteur par un exemple dans lequel deux transformations de modèles sont enchaînées (voir figure 58). Cet exemple de processus permet de transformer un modèle UML *mymodel*, chargé à l'exécution, en un modèle Python. Le résultat est enregistré sur le système de fichiers dans le dossier *myfolder*. Le nom du fichier résultat est généré automatiquement par le moteur.

Dans notre approche, les transformations sont considérées comme des activités à exécuter par le moteur de **PMS+**. La transformation *UMLToEcore* prend un paramètre d'entrée *source* de type métamodèle UML et un paramètre de sortie *target* de type métamodèle Ecore. La transformation dans ce cas permet de transformer un modèle UML,

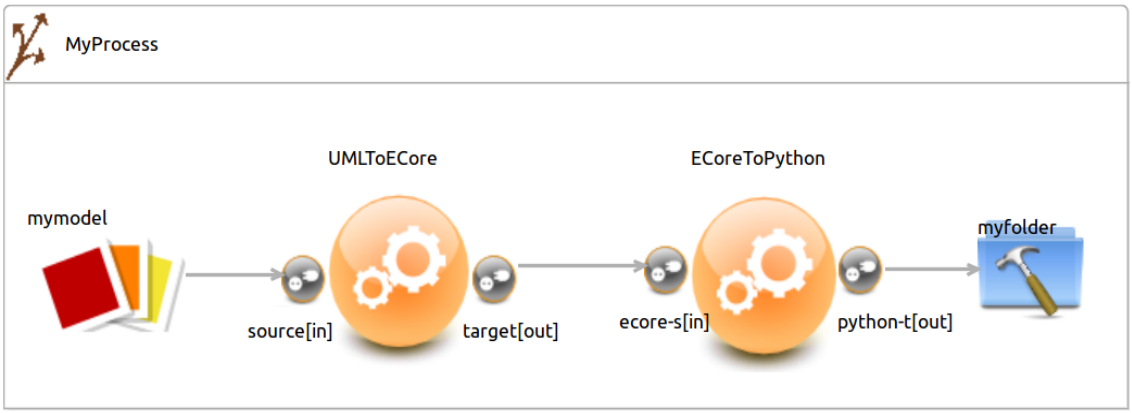


FIGURE 58: Exemple d'un processus de production

fourni en entrée, en un modèle Ecore. La transformation *EcoreToPython* a un paramètre d'entrée *ecore-s* de type métamodèle Ecore et un paramètre de sortie *python-t* de type métamodèle Python. Cette transformation permet de transformer un modèle Ecore, fourni en entrée, en un modèle Python. Le paramètre *source* est connecté à la donnée *mymodel* qui présente le modèle d'entrée UML. La donnée *mymodel* permet de charger le modèle afin d'être utilisé comme un objet et non comme un fichier. Le paramètre *ecore-s* est connecté au paramètre de sortie *target*. Enfin le paramètre *python-t* est connecté à la donnée *myfolder* qui présente le dossier dans lequel nous voulons enregistrer le résultat du processus. La donnée *myfolder* permet de référencer un dossier existant sur le système de fichiers.

Notre processus est complet. Nous pouvons à présent l'exécuter. Le moteur de **PMS+** commence par identifier les transformations dont tous les paramètres d'entrées sont présents. Dans notre cas, le moteur commence par exécuter la transformation *UMLToEcore* puisque son paramètre d'entrée *source* est présent et prend la valeur de la donnée *mymodel*. Le paramètre d'entrée *ecore-s* de la transformation *EcoreToPthon* n'est pas encore présent puisque celui-ci est conditionné par la présence du paramètre *target*. La donnée *mymodel* transite sur la connexion pour atteindre le paramètre *source* ensuite la transformation *UMLToEcore* est déclenchée. Celle-ci produit un modèle de sortie sur le paramètre de sortie *target*. Cette donnée transite sur la connexion pour atteindre le paramètre *ecore-s*. De ce fait, les paramètres d'entrée de la transformation *EcoreToPython* sont présents ce qui permet de la déclencher. Cette transformation produit un modèle de sortie sur le paramètre *python-t*. Cette donnée sera ensuite envoyée sur la donnée *myfolder* ce qui permettra de sauvegarder le modèle résultat sur le système de fichier dans le dossier référencé.

Dans la section suivante, nous présentons plus en détail les principaux concepts de **PMS+**.

7.2 PRINCIPAUX CONCEPTS

Les principaux concepts de notre approche sont les **activités**, les **données** et les **flux de données**. Le processus de production est décrit comme un graphe orienté dont les nœuds sont des **activités** ou des **données** et les arcs sont des **flux de données**. Ces derniers permettent de connecter les données aux paramètres des activités, considérés comme les points de connexion des activités, ainsi que les paramètres entre eux.

7.2.1 *Activités*

Dans **PMS+** les **activités** sont les unités à exécuter. Elles peuvent être une *transformation de modèle*, une *génération de code*, un *moteur d'exécution*, une *boîte noire* ou une *référence vers un autre processus de production*.

Une **activité** prend des paramètres d'entrée et/ou de sortie. Ces paramètres sont typés et ont une multiplicité. Les paramètres représentent ce que l'activité attend ou produit comme données.

Les activités de transformations de modèles et de génération de code représentent, comme leurs noms l'indiquent, une transformation ou une génération de code.

Une activité de type *moteur d'exécution* permet d'exécuter dynamiquement une ou plusieurs activités données en paramètres. Par exemple, dans un processus de production, le moteur peut être amené à générer une activité et ensuite à l'exécuter. Par exemple, une transformation abstraite (voir Domain Specific Transformation Language, DSTL, décrite dans la section 6.6.1), est accompagnée par une transformation souvent appelée HOT (Higher-Order Transformation). La HOT transforme la DSTL en une transformation **T** concrète séparée des modèles d'entrées (les modèles sur lesquels la transformation **T** doit s'appliquer). La transformation résultat **T** est ensuite exécutée avec ses modèles d'entrées. Dans ce cas de figure, nous avons un processus de production avec deux activités ; la première est la transformation HOT qui prend en entrée le modèle de la transformation abstraite, et la deuxième est une activité de type *moteur d'exécution* permettant d'exécuter la transformation résultat, considérée comme une activité.

Les activités de type boites noires permettent de réaliser des tâches autres que les tâches de base définies dans le moteur, par exemple faire appel à un service ou à une application externe. Dans **PMS+** un processus de production est lui-même considéré comme une activité, permettant ainsi une construction hiérarchique des processus de production.

7.2.2 *Données*

Les **données** représentent les objets véhiculés entre les activités. Nous pouvons spécifier des données primitives ou complexes. Pour les données complexes, nous pouvons spécifier des ressources permettant de référencer des éléments présents sur le système de fichiers, par exemple un fichier ou un dossier. Pour les données primitives, nous pouvons spécifier des entiers ou des booléens.

Dans un processus de production, il arrive souvent que le processus ait besoin de récupérer des données en fonction des résultats des activités exécutées. Pour cela, nous avons le concept d'*expression* permettant, à l'aide d'une expression OCL, d'exprimer une requête ou un filtre sur les données. Les expressions sont calculées dynamiquement à l'exécution du processus. *L'expression* permet entre autres de renvoyer le résultat d'une requête sur un modèle fourni par une activité. Par exemple, elle permet d'obtenir tous les éléments dont le type est "Class".

PMS+ propose un concept de *donnée utilisateur*. Ce concept permet de spécifier des interactions avec l'utilisateur, comme la saisie de données. Ce concept est une donnée, et est donc représenté comme un nœud dans le graphe du processus. Ce nœud est relié à un ou plusieurs paramètres d'une ou plusieurs activités. Lorsqu'une de ces activités est prête à être exécutée (sans tenir compte des paramètres reliés à la donnée utilisateur), **PMS+** propose une boîte de dialogue permettant la saisie des valeurs pour tous les paramètres reliés à la *donnée utilisateur*. Le moteur génère dynamiquement la boîte de dialogue en se servant des métadonnées des paramètres par exemple son type, son nom ou sa description.

7.2.3 *Flux de données*

Les flux de données sont orientés, ils sont représentés par un lien entre deux paramètres, ou un lien entre un paramètre et une donnée dans le diagramme. La direction du flux indique le sens de circulation des données. La source et la destination d'un flux de données doivent avoir le même type. Par exemple, un modèle peut être connecté à un paramètre d'entrées de type métamodèle d'une activité. Cela indique que les données correspondantes au modèle iront du nœud modèle vers le paramètre.

Un **flux de données** implique une dépendance de données entre la source et la destination (on dit alors que la destination dépend de la source). Dans le cas où la source et la destination sont des paramètres, le flux de données implique une dépendance entre les activités de ces paramètres. L'activité source est alors exécutée avant l'activité destination. Les flux de données permettent au moteur de **PMS+** d'ordonnancer les différentes activités du processus. D'une

manière générale, l'ordonnancement des activités est calculé selon les dépendances des données existant entre les paramètres. Dans **PMS+**, la notion de dépendance des données est déduite des flux de contrôle. Il arrive parfois que deux activités aient une dépendance entre elles qui n'est pas matérialisée par une dépendance de données. Pour résoudre ces cas, **PMS+** permet d'exprimer explicitement une dépendance entre deux activités. Une fois l'ordonnancement fait, le moteur commence par exécuter les activités dont tous les paramètres d'entrées sont présents. Les activités suivantes sont exécutées dès que leurs paramètres d'entrée sont à leur tour présents.

Les *flux de données* peuvent être contrôlés par des *gardes* à l'aide d'expressions booléennes indiquant si la donnée peut circuler sur le lien. Si l'expression est évaluée à vrai, la donnée peut aller de la source à la destination. Si l'expression est évaluée à faux la donnée est bloquée et par conséquent l'activité suivante ne sera pas activée. Ceci nous permet d'exprimer des contraintes bien définies sur le flux d'exécution.

Dans la section suivante, nous décrivons le métamodèle de **PMS+**, permettant de mettre en œuvre les concepts que nous venons de voir.

7.3 MPMS : LE MÉTAMODÈLE DE **pms+**

Le métamodèle MPMS définit les concepts permettant de décrire un processus de production de logiciels. Les modèles décrits à l'aide du métamodèle MPMS sont ensuite interprétés par le moteur d'exécution de **PMS+** afin d'être exécutés.

Dans **PMS+**, nous avons privilégié la réutilisation des activités. Pour cela, une activité est en fait une instance d'une *description d'activité*. Par exemple, la figure 59 montre les implémentations à gauche, les descriptions des activités (au centre) et leur utilisation (à droite). Cette technique est, aussi, adoptée par l'outil UniTI [145].

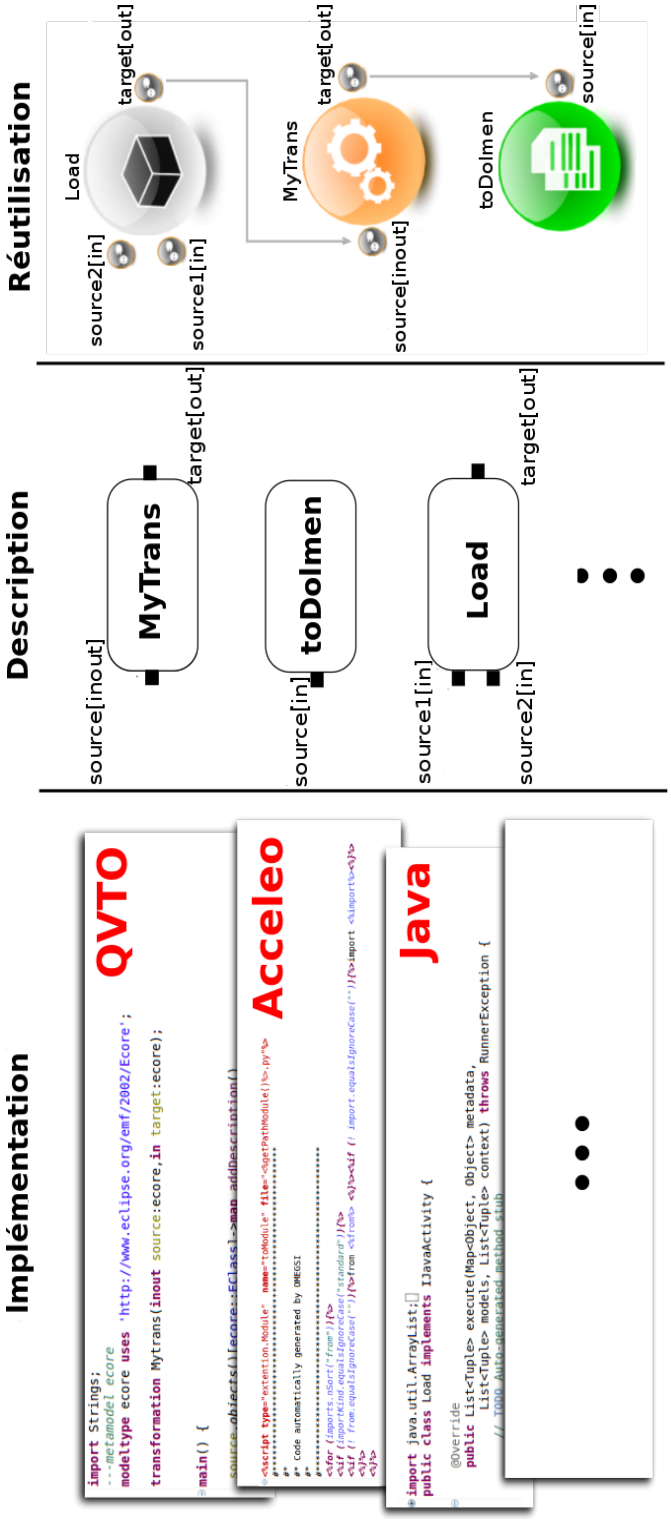


FIGURE 59: La réutilisation des implémentations dans **PMS+**

Ceci implique que pour pouvoir ajouter une activité dans un processus, il faut commencer par déclarer sa *description*.

Notre métamodèle est décomposé en trois parties (voir partie haute de la figure 60) ; la première partie est le cœur qui définit la notion de

graphe de manière abstraite et réutilisable ; la deuxième partie permet de décrire un processus de production en tant que graphe de nœuds d'activités, une activité étant ici une instance d'une *description d'activité* ; enfin la troisième partie sert à exprimer les *descriptions d'activités*.

Dans une telle décomposition, l'utilisateur commence par définir ses *decriptions d'activités* à l'aide du métamodèle de description des activités. Ces descriptions sont alors disponibles dans l'outil **PMS+**. Ensuite, il définit son processus en le composant avec des instances des descriptions d'activités.

Dans ce qui suit nous décrivons les différentes parties décomposant MPMS. Les détails de la figure 60 seront, ainsi, clarifiés.

7.3.1 *Métamodèle du cœur (core)*

Le métamodèle du cœur permet de capter les concepts abstraits qui servent de base pour les autres parties du métamodèle MPMS. Ces concepts permettent d'étendre facilement le métamodèle en ajoutant un nouveau concept métier. Par exemple, le concept *FlowGraph* voir figure 61) permet de décrire à un niveau abstrait ce qu'est un processus. Un *FlowGraph* a des *FlowNode* (nœuds) et des *Connection* (connexions) entre les éléments connectables *ConnectableElement*. L'élément connectable n'est pas toujours immédiatement le *FlowNode*, il peut être un élément composant le *FlowNode*. Par exemple dans le cas d'une activité, les éléments connectables sont les paramètres.

Cette description, à un niveau abstrait, permet au développeur de rajouter d'autres concepts de type *FlowGraph* au métamodèle, par exemple le concept de sous-processus. Le cœur permet aussi de simplifier les transformations de modèle en traitant les concepts abstraits. Les concepts du métamodèle du coeur ne sont pas accessibles à l'utilisateur final et ne sont utiles que pour les développeurs.

7.3.2 *Métamodèle des processus de production de logiciels*

Un processus de production est un graphe de *FlowNode* que nous représentons par le concept de *ProcessDesc*, sous type de *FlowGraph*. Les noeuds sont connectés entre eux par des connexions orientées qui peuvent être de type *SequenceEdge* ou *DataDependency*. (*SequenceEdge*) permet de décrire des connexions dont les gardes sont introduites dans le paramètre *guard* sous la forme d'une expression OCL, tandis que *DataDependency* décrit une dépendance de données entre deux activités qui sont elles-mêmes des *FlowNode*.

Les *Activity* (voir figure 63) constituent un type de noeud possible du graphe. Elles sont une référence (ou instance) d'une description d'activité (voir figure 60) enregistrée dans **PMS+**. Une activité peut être une instance de description de transformation de modèle

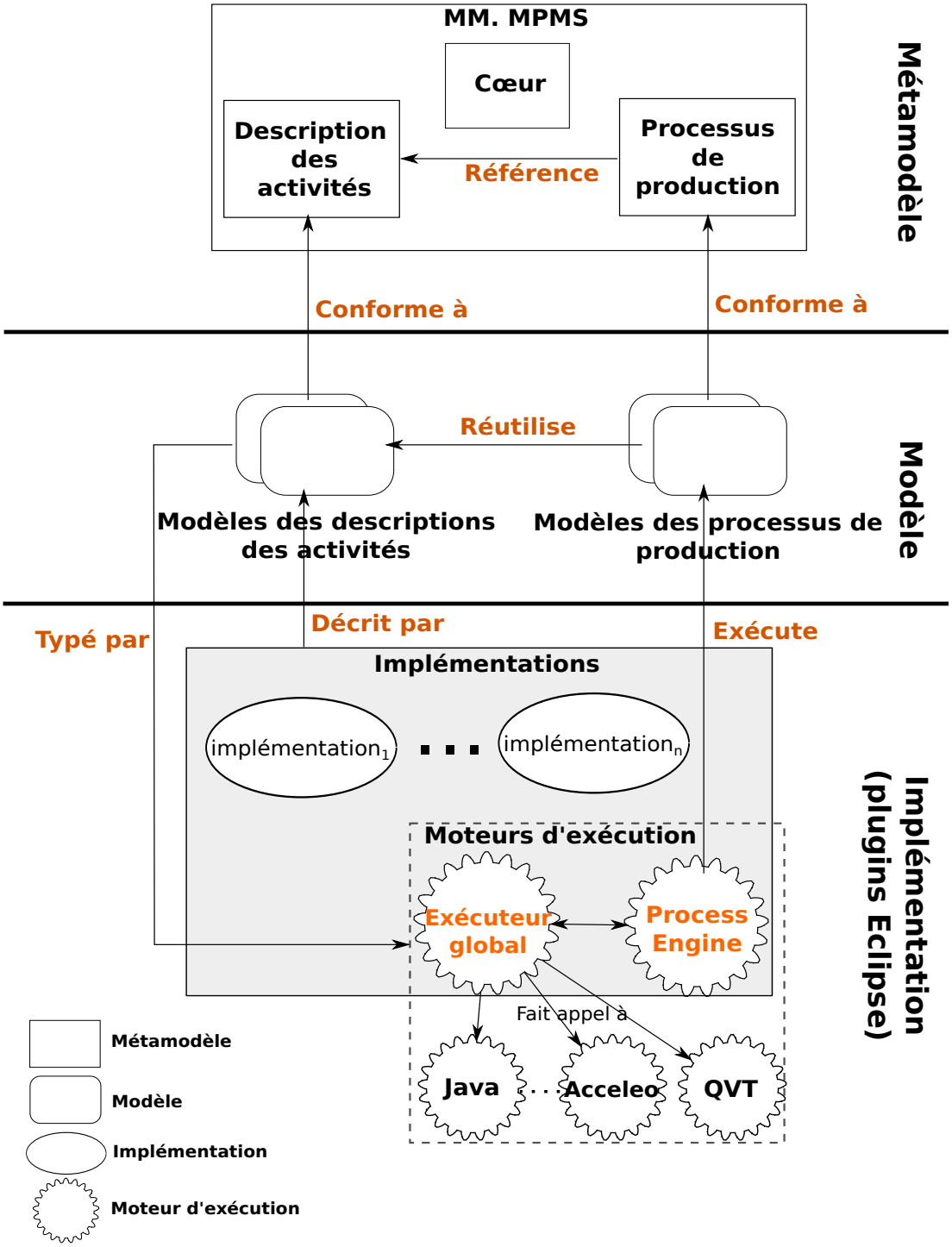


FIGURE 60: L'architecture de **PMS+**

ou de description de génération de code. Pour chaque type d'activité, le métamodèle propose les descriptions des activités de type correspondant, par exemple les descriptions des transformations pour les transformations.

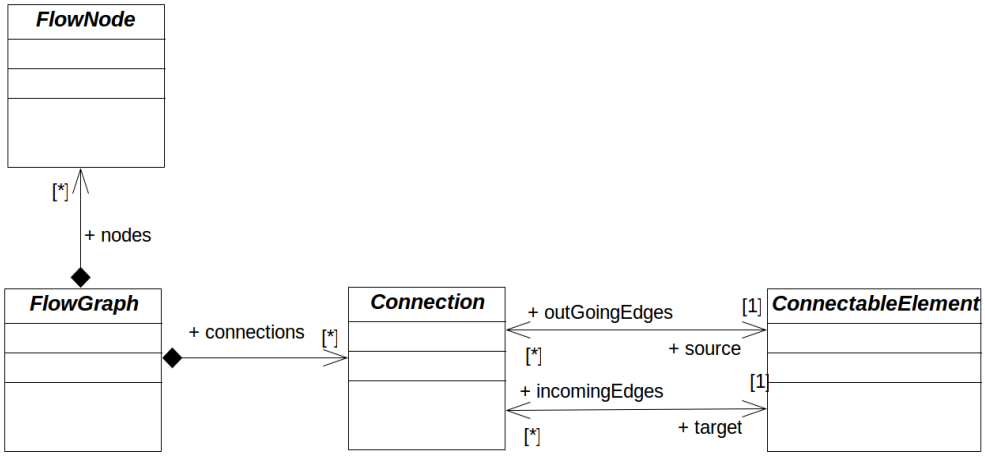


FIGURE 61: Une partie du metamodèle du cœur

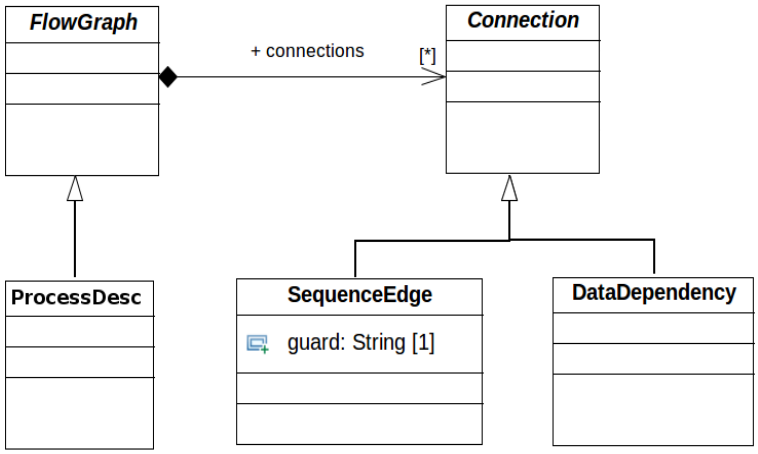


FIGURE 62: Metamodèle des processus de production

Les activités possèdent des paramètres *Parameter* qui référencent les descriptions des paramètres de la description de l'activité instanciée. Les paramètres servent de point de connexions avec les données ou avec d'autres activités. Pour ce faire, *Parameter* étend le type *ConnectableElement*.

Dans le modèle, l'utilisateur ajoute un nœud *Activity* puis choisit la description de l'activité à référencer dans le paramètre *description* ce qui provoquera le chargement automatique de la description, l'initialisation du nom de l'activité ainsi que la construction des paramètres automatiquement. Le changement de la description référencée provoque le chargement automatique de la structure de l'activité. Les paramètres constituent les points de connexion de l'activité. De ce fait, ils sont considérés comme des éléments connectables *ConnectableElement*.

Dans MPMS, les données sont décrites par le concept abstrait de *ValueSpecification* (voir figure 64). Ce concept est à la fois un élément connectable et un noeud du graphe. Le fait d'être un élément con-

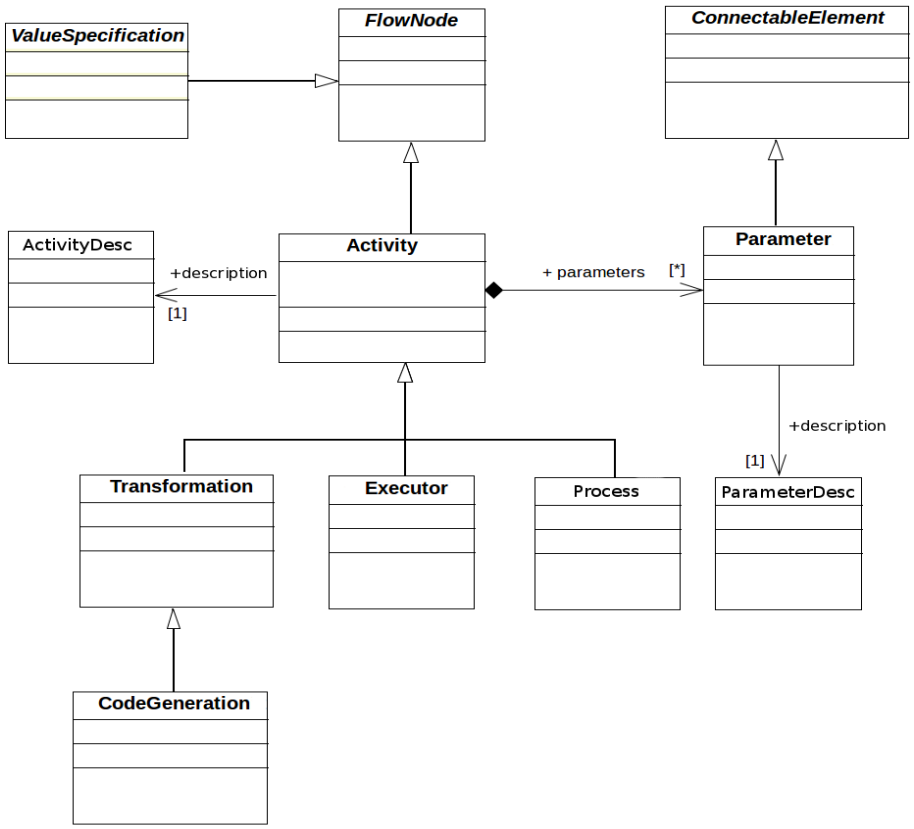


FIGURE 63: Métamodèle des instances d'activité

nectable permet de connecter la donnée à un paramètre d'une activité.

Parmi les données, le concept concret d'*Expression* permet de décrire les données calculables. Dans ce cas l'expression est introduite sous forme de chaîne de caractères dans le paramètre *expression* de ce concept. Nous avons aussi les *RessourceInstance* permettant de décrire les fichiers (*FileInstance*), les dossiers (*FolderInstance*) ou les modèles (*Model*). Ces derniers peuvent avoir une *uri* (l'adresse d'une ressource dans le système de fichiers) dynamique (sous la forme d'une expression calculée à l'exécution) permettant ainsi de référencer des ressources dynamiquement (par exemple l'URI d'un dossier produit par un générateur de code).

Pour une donnée utilisateur *UserSpecification* le moteur commence par transformer le paramètre auquel la donnée est connectée en un modèle décrit par notre métamodèle UI. Ce modèle décrit une interface graphique basée sur les formulaires. Ensuite, le moteur demande au modèle obtenu de renvoyer son rendu graphique permettant à l'utilisateur de saisir la donnée demandée. Une *UserSpecification* peut être connectée à plusieurs paramètres. Dans ce cas, une interface globale permettant de saisir toutes les données est générée.

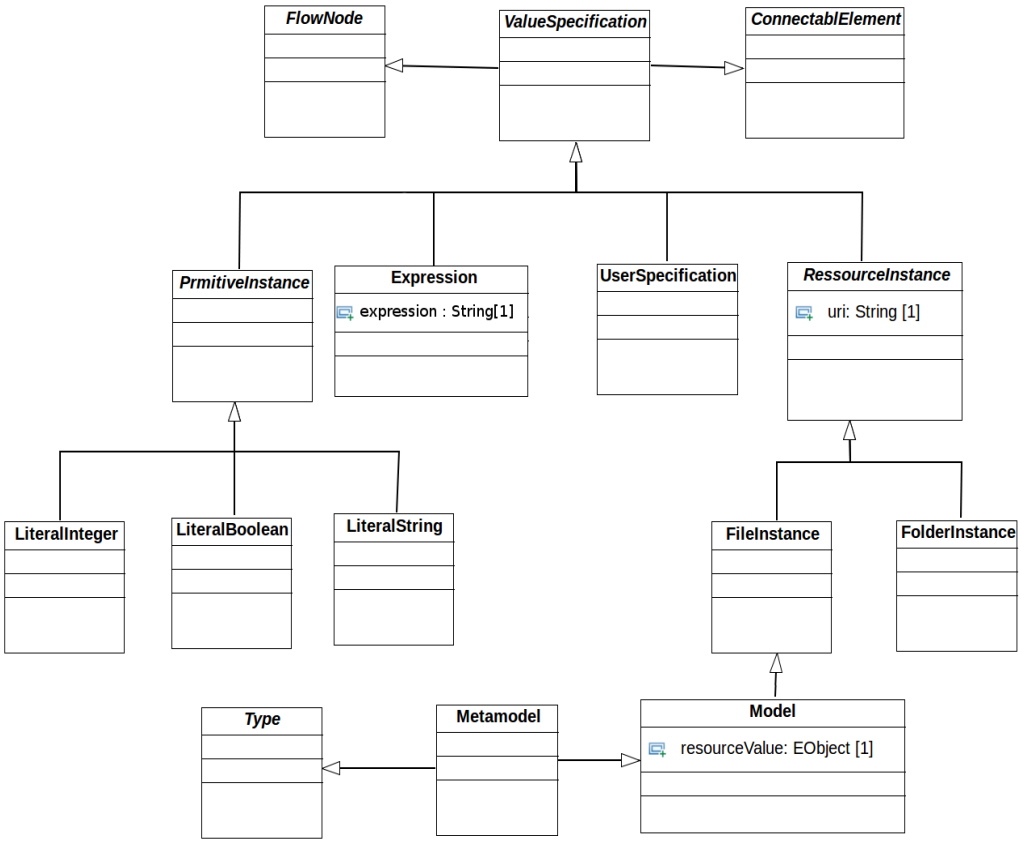


FIGURE 64: Métamodèle des données

7.3.3 *Métamodèle de description des activités*

Le métamodèle de descriptions des activités (voir figure 65) fait le lien entre les activités du processus de production et leurs implémentations dans une technologie. (voir figure 60). En effet, une description d'activité (*ActivityDesc*) permet de spécifier les paramètres requis et fournis par l'activité, ainsi que la technologie dans laquelle elle est implémentée. Une description d'activité peut par exemple être la description d'une transformation de modèle ou d'une génération de code. Les descriptions d'activités sont classifiées selon leurs fonctions : une description de type *TransformationDesc* pour les règles de transformation de modèle, de type *CodeGenerationDesc* pour les générations de code, etc. Une description d'activité comprend la description des paramètres *ParameterDesc* permettant de décrire ce que l'activité consomme et produit comme données. Le modèle de description des paramètres ainsi que celles des activités comportent un nom permettant de les identifier et une description (chaîne de caractères) permettant au concepteur de commenter l'activité. Ces données

sont utilisées lors de la génération des interfaces graphiques lors des interactions avec l'utilisateur.

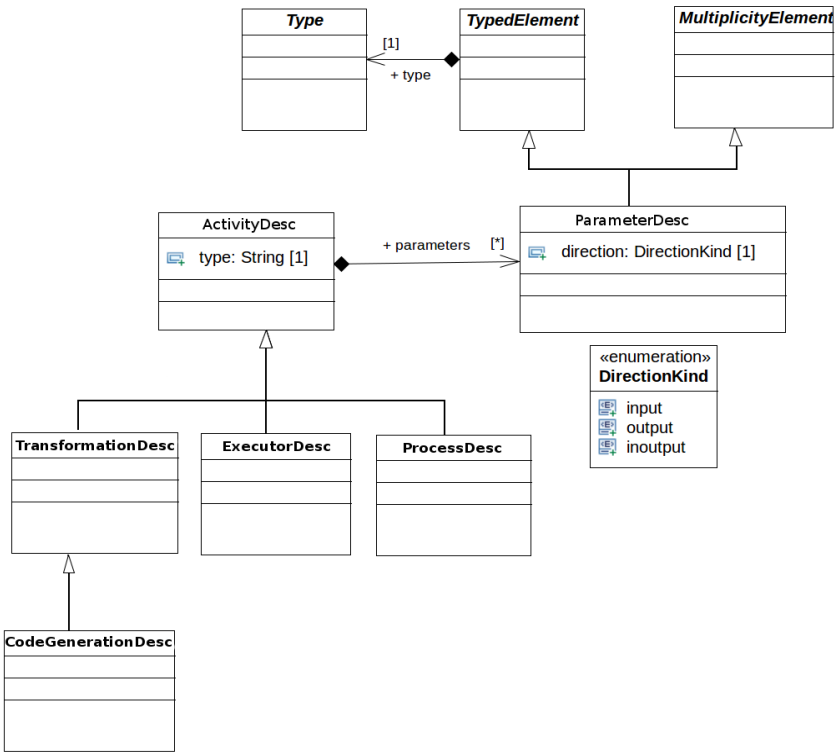


FIGURE 65: Métamodèle des activités

Les concepts du métamodèle ne comportent aucune référence à une technologie d'implémentation de l'activité. La spécification de la technologie se fait dans le modèle, à l'aide du paramètre *Type*. Ainsi, l'ajout d'une nouvelle technologie ne nécessite aucune modification du métamodèle. **PMS+** propose une librairie de technologies comportant déjà des implémentations pour QVTo [38], Acceleo [34] et Java. Il est possible d'ajouter de nouvelles technologies (voir 7.4.2) qui seront détectées automatiquement par **PMS+**. Une technologie est associée à son moteur d'exécution prenant en charge l'exécution de cette technologie. Les moteurs d'exécution peuvent être déclarés en tant que description d'activité de type *ExecutorDesc*. Par exemple, dans **PMS+** le moteur d'exécution des processus de production est écrit en QVT [105] et est déclaré en tant qu'*ExecutorDesc* de type QVT (voir figure 60).

La section suivante présente les particularités techniques de **PMS+**.

7.4 LES POINTS TECHNIQUES PARTICULIERS

PMS+ est implémenté par un ensemble de plugins Eclipse. L'outil est développé suivant une démarche basée sur la réutilisation. Ainsi, la plupart des fonctionnalités de **PMS+** sont décrites sous forme de

processus de production qui sont exécutés par le moteur d'exécution. L'ajout de nouvelles fonctionnalités peut se faire par ajout de nouveaux processus. Cette technique basée sur la réutilisation nous permet de maintenir ainsi que de faire évoluer **PMS+** plus facilement.

Les activités dans **PMS+** peuvent utiliser des technologies différentes , par exemple QVT, ATL[35], Java, etc. Si une technologie n'est pas supportée, **PMS+** permet de l'ajouter : il faut pour cela fournir le moteur d'exécution correspondant sous la forme de classe Java implémentant certaines interfaces. Le système d'ajout est basé sur le concept de point d'extension d'Eclipse [36]. Actuellement, **PMS+** supporte les moteurs d'exécution de QVT, Acceleo, Java et celui des *processus de production*.

Dans ce qui suit nous montrons comment ajouter une activité ainsi qu'une technologie dans **PMS+**.

7.4.1 *Comment ajouter une description d'activité*

L'ajout d'une description d'activité est le fait de la sauvegarder dans le registre des descriptions de **PMS+**. Ce registre permet de garder le modèle décrivant l'activité. Dans **PMS+**, la sauvegarde d'une description peut prendre deux formes ; l'installation et l'enregistrement temporaire.

L'installation implique que la description est validée par le concepteur et peut être intégrée dans un projet (par exemple, une application Eclipse). L'installation peut être faite dans le but de l'exporter définitivement dans un projet. Dans ce cas, elle est faite en packageant l'implémentation dans un plugin Eclipse. Ce plugin doit avoir un point d'extension permettant de déclarer l'implémentation en tant que description d'activité dans **PMS+**. Le point d'extension demande un modèle décrivant l'implémentation (le modèle de description de l'activité). Selon ce modèle l'implémentation est installée en tant que description de transformation de modèle, de génération de code ou d'exécuteur... Le type d'implémentation ne peut pas être déduit automatiquement d'où la nécessité d'avoir le modèle de description de l'activité.

PMS+, propose une fonctionnalité prenant en charge la génération du plugin et du modèle de description de l'activité correspondant à partir d'une implémentation. Cette fonctionnalité prend en paramètres le type de la description désirée ainsi que le fichier contenant l'implémentation. Elle explore le fichier afin de déterminer les métadonnées associées à l'implémentation, par exemple le nom ou les paramètres. Actuellement, la génération du modèle de la description dépend de la technologie de l'implémentation. En effet, certaines technologies, par exemple Java, sont difficiles à explorer automatiquement.

PMS+ propose, aussi, l'installation au niveau du runtime. Dans ce cas l'activité est prise en compte, automatiquement, à chaque démarrage de l'application. Ce type d'installation est fait en sauvegardant l'emplacement de l'implémentation dans le système de fichier. De ce fait, le changement de l'emplacement de l'implémentation rend l'installation obsolète.

L'enregistrement temporaire, quant à lui, permet de sauvegarder les implémentations temporairement, le temps de l'exécution de l'environnement Eclipse. L'enregistrement temporaire est utilisé par exemple dans le but de tester et de valider les activités. L'enregistrement est perdu à l'arrêt de l'environnement. Avec ce type de sauvegarde, l'implémentation n'a pas besoin d'être packagé.

7.4.2 *Comment ajouter une technologie*

L'ajout d'une technologie permet à **PMS+** d'exécuter des activités implémentées dans cette technologie. Afin d'ajouter une technologie dans **PMS+** nous nous basons sur la notion des points d'extension d'Eclipse. Dans **PMS+**, une technologie est une classe Java implémentant l'interface *IExecutor*. Celle-ci décrit l'ensemble des fonctionnalités qu'un exécuteur doit respecter, par exemple la fonction *execute* qui permet d'exécuter l'activité. Le point d'extension quant à lui permet de capter les métadonnées, par exemple l'identifiant de la technologie (qvto, Acceleo, java, etc.), de cet exécuteur afin que **PMS+** le prenne en compte et le propose comme une technologie possible au niveau du modèle de la description de l'activité. L'exécuteur sera ainsi enregistré dans le registre des exécuteurs, et sera pris en compte par l'exécuteur global de **PMS+** (voir figure 60). Ce dernier permet au moteur d'exécution des processus de déléguer l'exécution au bon exécuteur selon le type de l'activité. Le type de l'activité correspond à l'un des identifiants des technologies présentes dans **PMS+**.

7.4.3 *Comment exécuter un processus de production*

Une fois le processus de production modélisé, il faut pouvoir l'exécuter. Si le processus nécessite des paramètres en entrée, il faut un mécanisme permettant de saisir ou de spécifier la valeur de ces paramètres. **PMS+** propose deux approches pour exécuter un processus : une approche généraliste et une approche personnalisée.

L'approche généraliste permet d'exécuter n'importe quel processus de production. Pour cela, l'utilisateur sélectionne l'action *Execute Process* du menu de **PMS+**. **PMS+** propose alors une boîte de dialogue dans laquelle l'utilisateur renseigne le modèle du processus à exécuter, ainsi qu'un modèle contenant les valeurs des paramètres nécessaires à l'exécution.

Dans l'approche personnalisée, l'utilisateur peut ajouter des actions personnalisées dans la barre d'actions d'Eclipse. Chaque action permet alors de déclencher le processus de production qui lui est associé. Les actions peuvent être installées dynamiquement, sans avoir besoin de relancer **PMS+** ou Eclipse. Pour installer une action personnalisée, il faut choisir la fonctionnalité *Create Process Action*. **PMS+** propose alors une boîte de dialogue dans laquelle l'utilisateur renseigne le modèle du processus à installer ainsi que le menu dans lequel ce processus doit apparaître. Le menu est décrit par une chaîne de caractères sous la forme *sousMenu1 : :... : :sousMenuN : :action*. L'utilisateur a ainsi un accès rapide, personnalisé à ses processus. Cette seconde méthode, quoique plus longue à mettre en œuvre, s'avère plus intéressante lorsque l'utilisateur est amené à exécuter souvent les mêmes processus de production.

Il est à noter que les actions *Execute Process*, *Create Process Action*, ainsi que la plupart des actions du menu de **PMS+**, sont elles-mêmes décrites par des actions personnalisées associées à des processus de production. L'utilisateur peut ainsi très facilement étendre les actions de **PMS+** en ajoutant des actions associées à des processus de production.

7.5 DE MACOP AU PORTAIL COLLABORATIF

Nous avons utilisé **PMS+** pour developper le processus de production nous permettant de générer le code Python des portails collaboratifs à partir de leurs descriptions faites en MACoP. Le processus de production nous permet de passer d'un espace de modélisation de haut niveau d'abstraction à l'espace de génération de code, considéré de bas niveau d'abstraction. Ce processus est composé de plusieurs niveaux de modélisations intermédiaires, chaque niveau permettant de descendre progressivement en abstraction. Les modèles intermédiaires sont décrits par des métamodèles permettant d'ajouter les détails techniques et technologiques nécessaires à la génération de code.

Nous avons pris soin de faire apparaître le plus tard possible les détails technologiques propres à la cible visée, ici Python. Ces détails n'apparaissent que dans les derniers modèles de la chaîne. Ils sont ajoutés automatiquement par les transformations sans aucune intervention humaine.

Les transformations sont écrites en QVT et captent les connaissances du maître d'œuvre pour l'analyse du modèle et celle du développeur pour les stratégies de développement. Elles sont réparties sur trois grands niveaux d'abstraction, chaque niveau étant lui-même découpé en plusieurs sous-niveaux de détails introduisant chacun des concepts associés au domaine (voir figure 66). Chaque grand niveau est décrit par un processus de production à part entière (processus de production métier, technique et technologique). Un quatrième proces-

sus de production se charge de l'enchaînement de ces trois processus. Nous trouvons ainsi :

- **Le processus de production métier** dans lequel, nous appliquons les différents aspects décrits dans le modèle d'entrée, puis nous (voir section 6.6.1) ajoutons les processus métier par défaut, permettant de manipuler les objets de collaboration pour lesquels le designer n'a pas défini de processus. Ainsi, le modèle est enrichi et complété (c'est-à-dire qu'il existe pour chaque information au moins un processus la manipulant) suivant des règles métier conventionnelles.
- **Le processus de production technique** dans lequel, nous transformons les objets de collaboration en classes avec leurs interfaces, et optimisons les expressions OCL en les transformant en des méthodes pour les objets de collaborations sous forme d'adaptateurs. Les actions sont transformées en classes munies de leurs formulaires suivant différents patterns de modélisation selon leurs contextes et implémentation. Les paramètres d'entrée des *implémentations* des actions ainsi que les contextes de réalisation sont transformés en formulaires. À ce niveau, nous mettons à plat les composants afin d'uniformiser les implémentations des comportements (action, fonctionnalités, etc).
- **Le processus de production technologique** dans lequel, nous transformons les expressions OCL en expressions Python, formons les corps (c'est-à-dire les blocs d'instructions) des méthodes, et transformons les composants externes en code intégré au modèle. Les instructions et méthodes de validation des formulaires et des attributs des classes sont ajoutées automatiquement. Il en est de même pour les différentes méthodes propres au système de sécurité. À la fin de ce processus, nous obtenons un modèle conforme à un métamodèle du langage Python que nous avons défini. À partir de ce modèle, nous générons le code Python qui sera associé aux librairies standard requises. Cela nous permet d'avoir une maîtrise du modèle allant jusqu'au niveau de l'instruction, ce qui nous est utile pour l'optimisation du code généré, le calcul des imports, etc. Dans certains cas, le modèle Python obtenu peut contenir des dépendances cycliques entre les modules. Afin de les résoudre, nous appliquons une transformation d'optimisation permettant de réorganiser les différentes classes.

Nous considérons la transformation *UMLToMACoP* comme une transformation d'ordre technique permettant de produire un modèle métier *target* décrit par le métamodèle MACoP ainsi qu'un modèle contenant les différents composants et aspects externes utilisés. La génération de code, dans notre cas, se limite à la transcription du contenu du modèle technologique et ne contient aucune connaissance d'ordre technologique. C'est pourquoi elle est mise à l'extérieur de la

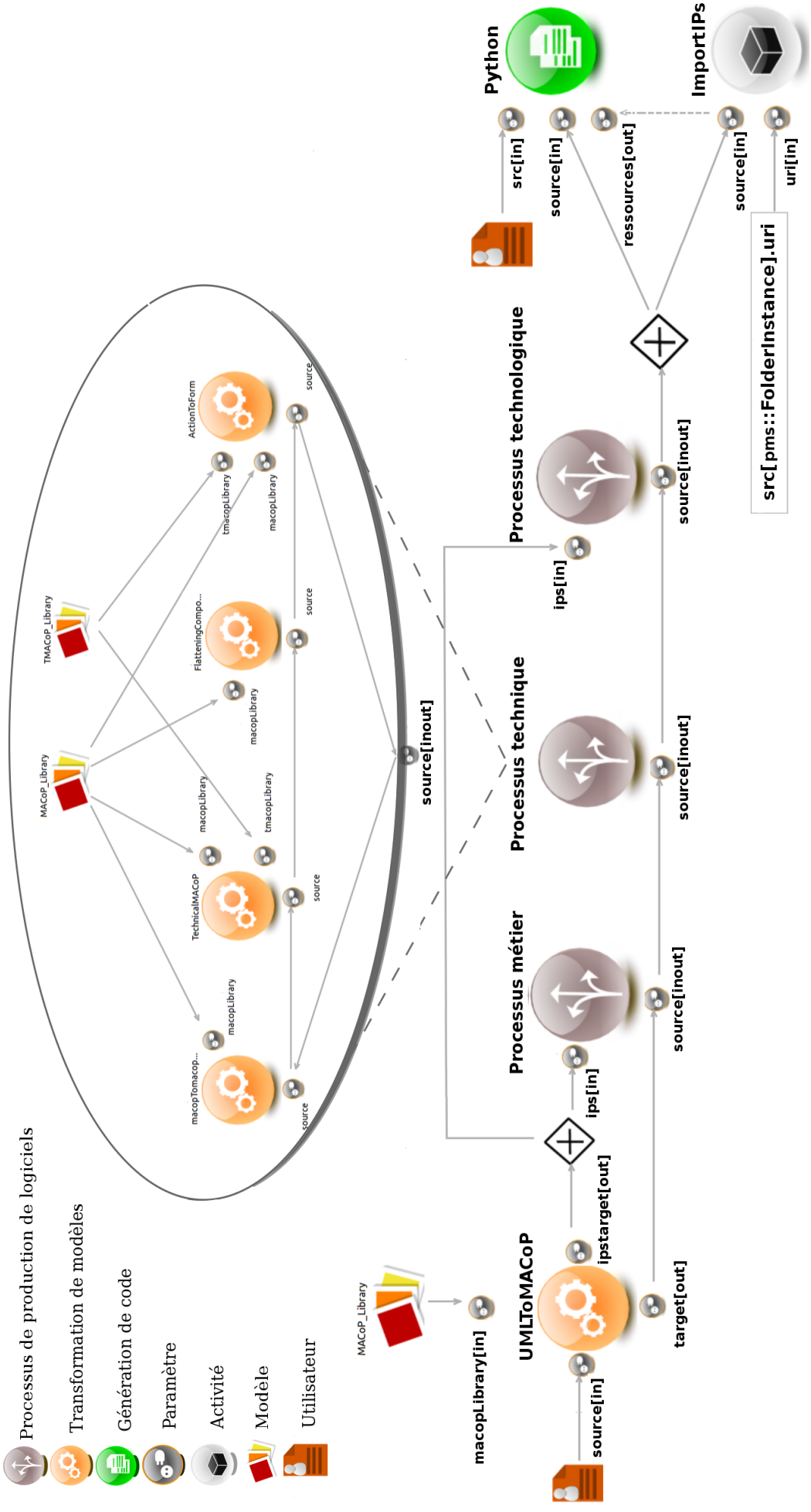


FIGURE 66: Processus de production MACoP->Portail Collaboratif(Python)

stratégie technologique. Enfin, l'activité *importIPs* permet d'intégrer le code des composants externes utilisés dans le code généré.

Le processus de production MACoP->Portail Collaboratif(Python) nous permet de générer 100% du code fonctionnel. Le développeur n'a plus qu'à lancer l'application. Le portail collaboratif généré permet de manipuler les données de la collaboration en conformité avec les processus métier spécifiés au niveau modèle. Les processus métier seront exécutés d'une manière implicite en agissant directement sur les objets avec lesquels ils ont une relation directe ou indirecte.

7.6 CONCLUSION

Dans ce chapitre nous avons présenté notre outil **PMS+** de modélisation et d'exécution des processus de production de logiciels dans une démarche IDM. Cet outil permet de décrire des processus de production complexes centrés sur le modèle. Le modèle dans ces processus est interrogeable. Les processus de production permettent l'enchaînement des activités suivant un flux d'exécution orienté selon des contraintes relatives aux données manipulées par ces activités. Ces dernières peuvent être des transformations de modèle, des générations de code ou des boites noires, etc. Dans **PMS+** nous avons simplifié la notion d'interaction avec l'utilisateur ainsi que les interactions avec le système de fichiers. Cela permet, par exemple, de charger ou d'enregistrer des données selon les choix de l'utilisateur. Pour cela notre outil génère automatiquement des interfaces utilisateurs basées sur les formulaires afin de saisir les données et les injecter dans le processus.

PMS+ est totalement indépendant des technologies utilisées par les activités. Le concepteur peut étendre **PMS+** en lui permettant d'interpréter de nouvelles technologies. Ces extensions se font à l'aide d'un mécanisme basé sur les points d'extension d'Eclipse. Les technologies actuellement supportées par **PMS+** sont QVT, Acceleo, Java et le moteur de processus. Les fonctionnalités de **PMS+** sont elles-mêmes décrites sous forme de processus de production réutilisables ce qui permet de les maintenir en utilisant **PMS+** lui-même.

Plusieurs travaux futurs sont envisagés pour **PMS+**. Par exemple, les expressions qui décrivent les données calculables et les gardes sont dépendantes du métamodèle MPMS. Afin de rendre leur écriture plus simple, il faut cacher cette dépendance. D'autres travaux concernant l'amélioration des diagrammes afin de rendre la modélisation plus ergonomique sont en cours.

L'outil **PMS+** nous a permis de modéliser et exécuter notre processus de production permettant ainsi de générer des portails collaboratifs à partir d'un modèle MACoP. Ce processus de production est, principalement, divisé en trois sous processus : le processus de production métier qui permet de compléter le modèle d'entrée ; le

processus de production technique qui permet d'introduire les concepts techniques comme les interfaces et les classes ; le processus de production technologique qui permet d'introduire les concepts technologiques à grain fin (c'est-à-dire au niveau de l'instruction). Ce dernier produit un modèle complet de la technologie cible ce qui permet d'appliquer des processus d'optimisation de code par exemple. Un quatrième processus nous permet de spécifier l'enchaînement des trois autres, et ainsi de générer le code de nos applications de portail collaboratif à partir de leur description MACoP.

Dans ce qui suit, nous allons voir un exemple d'utilisation du métamodèle MACoP.

Quatrième partie

VALIDATION EXPERIMENTALE

8 EVALUATION

Sommaire

Précédemment nous avons parlé du métamodèle MACoP nous permettant de modéliser nos portails collaboratifs. Cette modélisation est centrée sur les processus métier centrés sur les données. À partir d'un modèle conforme au métamodèle MACoP nous générons le portail collaboratif opérationnel correspondant. Pour la génération de code, nous utilisons l'outil PMS+ que nous avons développé. Cet outil nous permet de modéliser ainsi que d'exécuter nos processus de production de logiciels.

Dans ce chapitre nous évaluons notre approche à travers l'exemple du métier d'Ecréall (section 8.1). Ensuite dans la section 8.2 nous comparons notre approche par rapport aux autres approches de modélisation identifiées dans la section 4.3. Cette comparaison est faite selon les critères identifiés dans la section 5.2.

8.1 MODÉLISATION DU MÉTIER D'ECRÉALL

Aujourd'hui, Ecréall travaille sur un mode collaboratif avec ses clients à travers l'usage de différents outils dont le « Tracker ». L'usage de cet outil rentre dans le cadre d'une démarche agile de gestion de projet informatique. Le client y crée des tickets pour demander de nouvelles fonctionnalités ou pour signaler des anomalies dans le logiciel livré. Les tickets sont regroupés par itérations, qui sont des sessions de développements avec un nombre de jours-homme fixe. Chaque ticket possède un état parmi « Non confirmé », « Ouvert », « En recette » ou « En production ». De ce fait, à tout moment, le client peut visualiser l'état d'avancement de son projet en fonction des états des tickets. Pour chacun de ses clients, Ecréall met en place

un « Tracker » sous forme d'application web. Cet outil ne gère pas les processus de facturation ni la gestion des réunions.

Le « Tracker » est l'outil utilisé jusqu'ici par Ecréall. Nous voulons le moderniser en modélisant le métier correspondant, et en ajoutant de nouvelles fonctions. Cette modélisation est faite à l'aide de MaCOP.

Afin de structurer et d'uniformiser le métier d'Ecréall, nous avons modélisé une partie de son métier pour enfin générer le portail collaboratif correspondant. Dans l'application générée, les clients d'Ecréall sont gérés dans la même application. Chaque client, dans ce cas, a son propre espace de travail. Avant de pouvoir utiliser l'application générée, un client doit s'enregistrer auprès de l'application. Dans cette section nous montrons les différents modèles décrivant le métier d'Ecréall selon les vues informationnelles, les vues des ressources et les vues fonctionnelles. Puis nous détaillons l'implémentation à base de composants de quelques actions des processus métier modélisés. Ensuite, nous illustrons un exemple de modélisation par aspect. Cet aspect concerne la gestion des réunions dans Ecréall. Enfin, nous montrons quelques captures d'écrans présentant l'application générée à partir du modèle du métier d'Ecréall obtenue.

8.1.1 *La vue informationnelle*

Nous commençons la modélisation de notre application par la description des objets de collaborations illustrés aux figures 67, 68 et 69. Dans ces figures, les concepts en blanc correspondent aux objets de collaboration de type « Information » du métamodèle MACoP, les concepts en bleu correspondent aux profils de type « Human » et, enfin, le concept en gris correspond à l'objet de collaboration de type « Application ». Dans ce modèle, nous spécifions deux types de contraintes : les contraintes structurelles décrites par les relations entre les objets de la collaboration ; et les contraintes fonctionnelles sur chaque objet, décrites en utilisant le langage OCL.

La racine de l'application (c'est-à-dire objet de collaboration de type « Application » du métamodèle MACoP) prend la forme du concept « ApplicationEcreall » (voir figure 67), qui contiendra les espaces de travail de type « EspaceDeTravail ». Ceux-ci sont des espaces privés propres à chaque client. Les espaces de travail contiennent les projets de type « Projet » du client. Dans ce modèle, les utilisateurs seront représentés par des profils « Utilisateur » (C'est un objet de collaboration de type « Human » du métamodèle MACoP) qui seront stockés dans un container de type « ContainerUtilisateurs ».

Un projet contiendra les facturables de type « Facturable » (voir figure 68) par exemple les audits de type « Audit », les formations de type « Formation » ou les itérations de type « Iteration ». Dans le cadre de la logique métier de l'application Tracker, des tickets de type

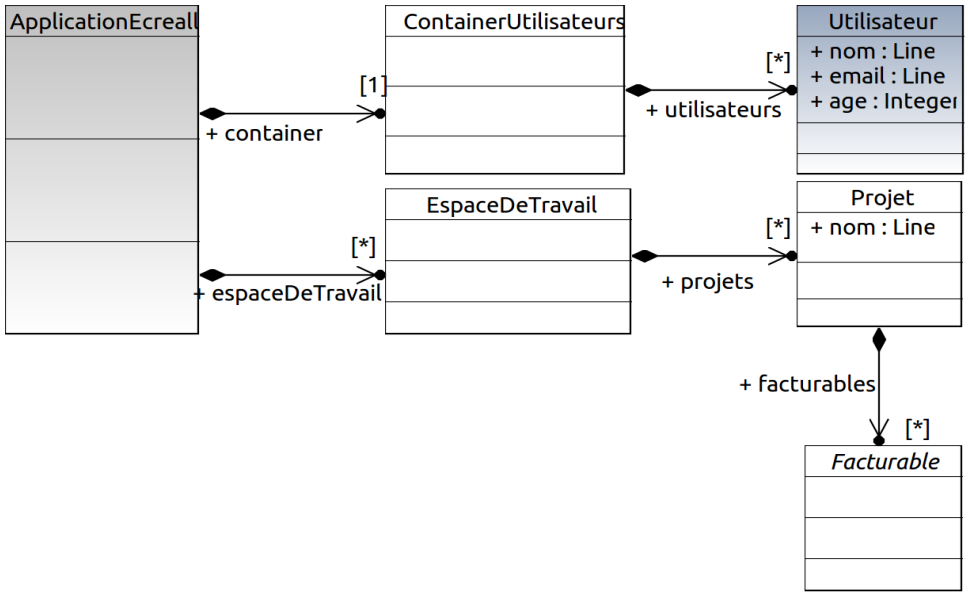


FIGURE 67: Modèle de la vue informationelle « ApplicationEcreall »

« Ticket » seront associés aux itérations. Ceux-ci peuvent être commentés grâce au type « Commentaire ». Lors de sa création, un ticket est assigné par défaut à une itération spéciale de type « Backlog ». Chaque ticket à une complexité (de type entier naturel) permettant de donner une estimation de la difficulté associée au ticket.

Les espaces de travail, les projets et les facturables héritent du type « PorteDocument » (voir figure 69), qui permet d'associer des documents de type « Document ». Ces derniers peuvent être des fichiers de type « Fichier » ou des dossiers de type « Dossier ». Nous trouvons, aussi, la notion de réunion « Reunion ». La relation de celle-ci avec l'application n'est pas spécifiée d'une manière directe. Afin de décrire les réunions, nous lui avons appliqué l'aspect « Gestion des réunions » que nous expliquerons plus loin (section 8.1.5).

Dans notre modèle, nous pouvons ajouter des contraintes et des invariants, par exemple un projet ne peut avoir qu'une seule itération de type « Backlog ». Cet invariant est décrit sur l'objet de collaboration « Projet » sous la forme d'une expression OCL :

« **self.facturables[Backlog]->size()<=1** »

Une fois le modèle des objets de collaboration défini nous pouvons configurer l'application. Ici, nous avons créé l'instance de notre application « applicationEcreall ». Nous créons l'unique instance du « ContainerUtilisateurs » nommée « containerutilisateurs » qui mémorisera les profils de l'application (voir figure 70). Ainsi au démarrage de notre application, nous aurons le nœud racine contenant le nœud « containerutilisateurs ».

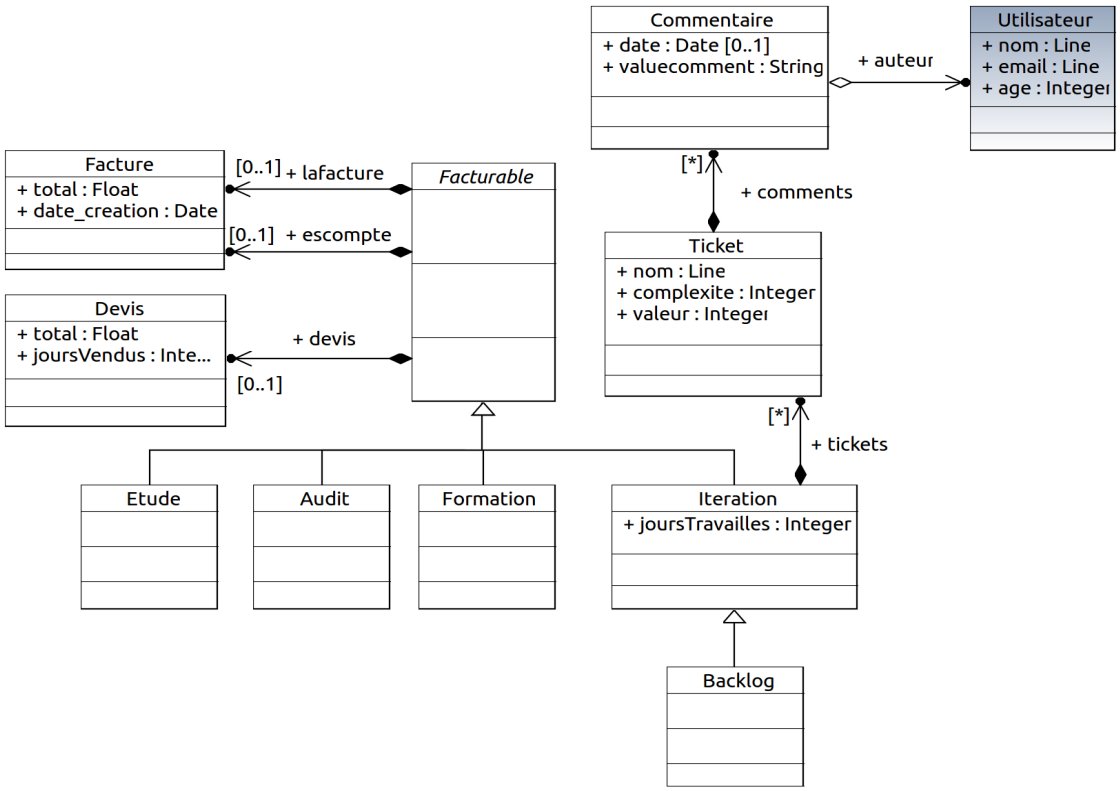


FIGURE 68: Modèle de la vue informationelle « Facturable »

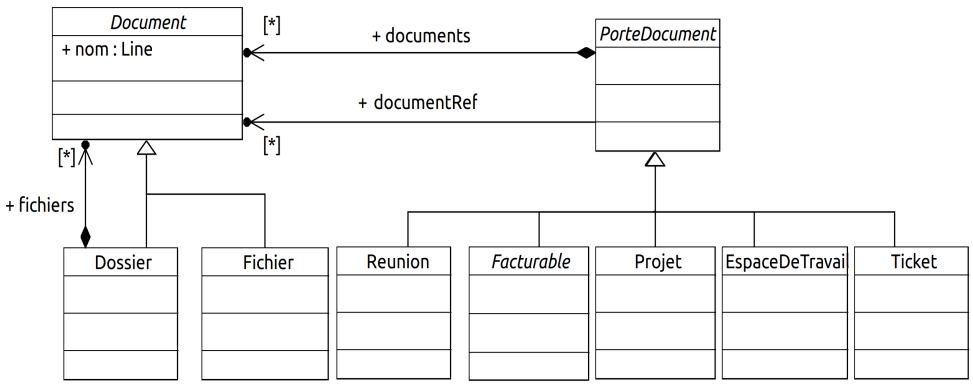


FIGURE 69: Modèle de la vue informationelle « PorteDocument »

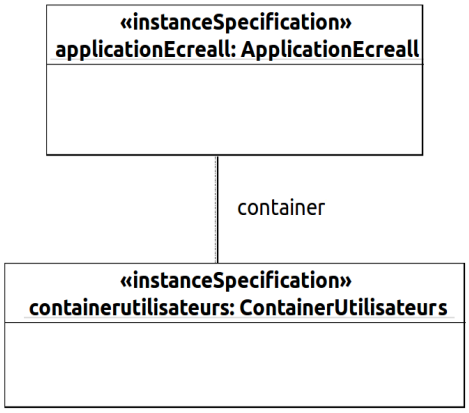


FIGURE 70: Modèle de l'initialisation de l'application

8.1.2 *La vue des ressources*

Après la modélisation de la partie statique de l'application (c'est-à-dire le « Quoi ? »), nous modélisons le « Qui ? » et le « Comment ? » de notre application. En premier, nous définissons les rôles dans l'application, dans notre cas, nous avons les rôles à affecter globalement par rapport à l'application, illustrés à la figure 71, qui sont le client « Client », le développeur « Développeur » et enfin le chef d'équipe « Chef d'équipe » qui est le supérieur hiérarchique des développeurs. Et les rôles à affecter localement par rapport à une instance d'objet de collaboration, illustrés à la figure 72, qui sont le responsable du ticket « ResponsableTicket », le chef de projet « Chef de projet », le contributeur à un projet « Contributeur » et enfin le bénéficiaire du projet « Bénéficiaire ». Notons ici, la relativité de la portée des rôles dans notre modélisation. En effet pour les anciennes applications Tracker mises en place par Ecréall, nous avons un client par application. Le rôle « Client » et le rôle « Bénéficiaire » se confondent puisque le client est le bénéficiaire du projet. Par contre, notre nouvelle application est unique et contient tous les clients d'Ecréall. Une séparation entre le rôle « Client » et le rôle « Bénéficiaire » est donc obligatoire.

Dans la vue des ressources, nous décrivons également les domaines d'activité et les processus domaine (Processus métier) à un niveau de détail simplifié. Dans la figure 72, le domaine d'activité « Tracker » est composé des processus domaine « Gestion des tickets » et « Gestion des itérations ». Cela permet d'avoir une idée globale des aspects métier impliqués dans la collaboration. Dans la figure 71, le domaine d'activité « Suivies des projets » est composé du processus « Gestion des projets ». Ce processus est détaillé dans l'annexe B.2.

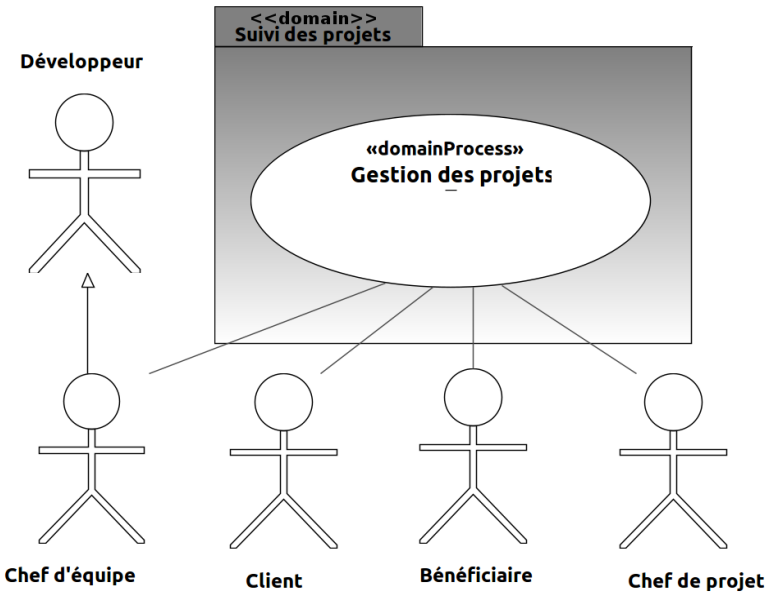


FIGURE 71: Le domaine « Suivi des projets »

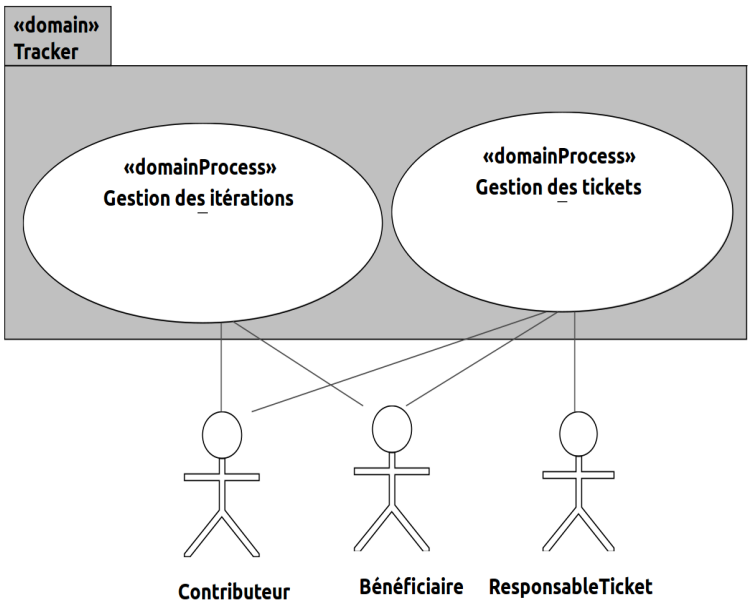


FIGURE 72: Modèle des rôles

8.1.3 *La vue fonctionnelle*

Nous devons maintenant raffiner chaque processus domaine en spécifiant le processus métier BPMN associé. Dans le modèle du métier d'Ecréall, certains objets de collaboration ne sont pas associés à des processus métier les manipulant. Par exemple, la gestion des utilisateurs n'est pas décrite dans ce modèle. Dans ce cas, notre processus de production de logiciels, servant à générer l'application, se charge d'appliquer l'aspect gestion des utilisateurs par défaut. De même pour les informations auxquelles l'utilisateur n'a pas spécifié de processus de gestion, le processus de gestion par défaut sera appliqué.

Dans ce qui suit, nous détaillons le processus « Gestion des tickets » et le processus « Gestion des itérations ». Nous détaillons, aussi, le processus de visualisation des espaces de travail « Visualisation de l'espace de travail » et nous donnons un exemple de processus imbriqué « Facturation des facturables ».

Le processus « Gestion des tickets »

La figure 73 détaille le processus métier « Gestion des tickets ». Celui-ci peut être initié par le Contributeur (Développeur appartenant à l'équipe de développement désigné dans le processus « Création des projets ») ou par le Bénéficiaire du projet (le client du projet) lors de la création d'un ticket. Cette action a comme contexte principal une instance d'« Iteration » (c'est-à-dire que l'action « Créer Ticket » sera accessible quand l'utilisateur étudiera une itération). Le contexte de production est le ticket créé et la cible est l'itération spécifiée par le contexte principal. Si un Contributeur est l'initiateur, il doit choisir un des Bénéficiaires du projet présents dans l'application comme ResponsableTicket de ce ticket (fait dans l'action « Créer Ticket »). Si le ticket est créé par un Bénéficiaire, il en devient le responsable (ResponsableTicket), un Contributeur pourra alors commenter, modifier la complexité ou commencer le développement (l'écriture du code) de ce ticket. À noter que les actions « Commenter » et « Modifier la complexité » sont des actions multi instance parallèle. Une action multi instance parallèle (voir l'annexe A.4) est une action proposée par le métamodèle BMPN indiquant que l'action peut être instancier plusieurs fois. Les instances de cette action peuvent être exécutées, parallèlement, par plusieurs personnes.

Une fois le développement fini, le contributeur met le ticket en recette en attente de validation du responsable du ticket. Le responsable du ticket, quant à lui, pourra assigner le ticket à une itération (si l'itération n'est pas dans l'état figée) appartenant au projet en question. Cette action est une action multi instance parallèle. Une fois que le ticket est en recette, le responsable du ticket pourra le valider afin que le développeur puisse produire les actions associées (démon-

stration, intégration dans l'application du client, etc.). Dans le cas où le ticket est refusé par son responsable, le développeur repasse à la phase de développement.

Le processus métier est associé à un data-store contenant des instances des objets manipulés (en relation avec le processus). Nous y avons spécifié des instances du « Ticket », de l' « Iteration » et du « Commentaire ».

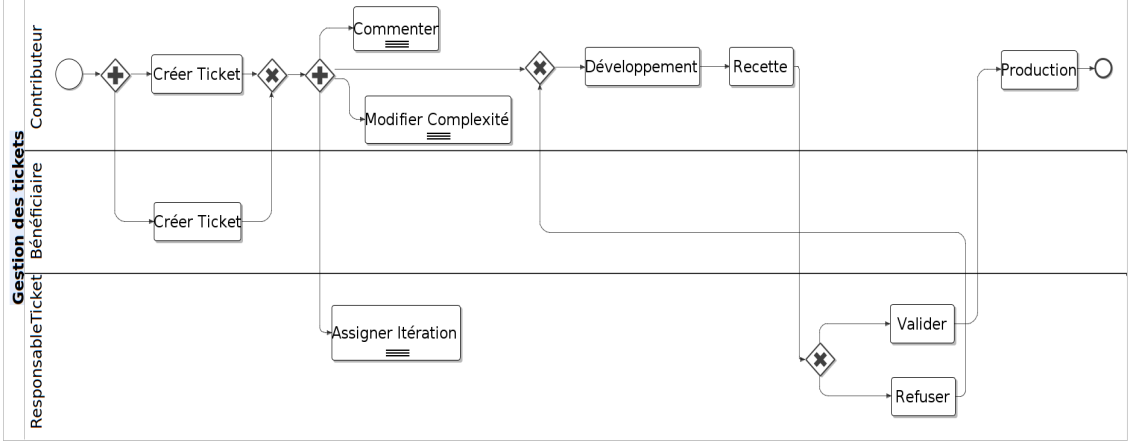


FIGURE 73: Le processus métier «Gestion des tickets »

Le processus « Gestion des itérations »

La figure 74 illustre le processus « Gestion des itérations ». Dans ce processus, un Bénéficiaire (un client bénéficiaire du projet) commence par créer une itération dans un projet de son espace de travail. Une fois que tous les tickets présents dans l'itération ont une complexité, un Contributeur (développeur du projet) pourra y créer le devis associé. Dans le processus métier « Gestion des itérations », le flux de séquences connectant l'action « Creer iteration » et l'action « Creer devis », est un flux synchrone (voir section 6.5.2). De ce fait la condition vérifiant si tous les tickets ont une complexité est réévaluée à chaque fois que le processus est demandé par l'application. Une fois le devis créé, le contributeur pourra le modifier ou l'envoyer, par courriel électronique, aux bénéficiaires.

Afin d'envoyer le devis, le contributeur a besoin de voir les différents tickets de l'itération avec leurs complexités et leurs états ainsi que les données associées au devis de l'itération. Ces données sont spécifiées dans le contexte de pré-étude de l'action (voir figure 75).

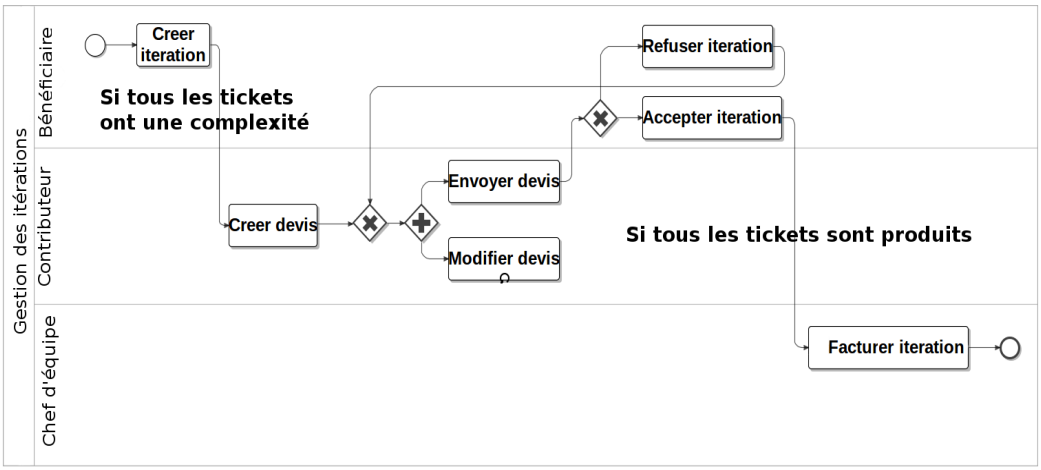


FIGURE 74: Le processus métier « Gestion des itération »

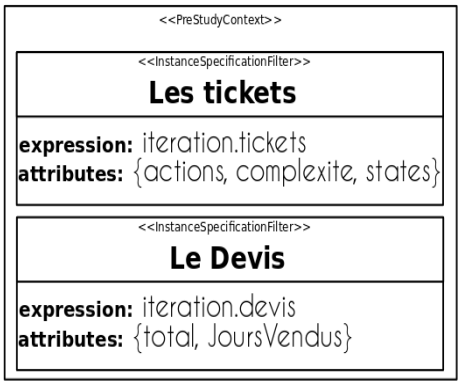


FIGURE 75: Le contexte local de pré-étude de l'action « Envoyer devis »

Une fois le devis envoyé, l'itération devient figée (les Responsables ne pourront plus lui assigner de tickets). Ensuite, le Responsable doit valider ou refuser l'itération. Si elle est refusée, elle devient à nouveau ouverte et les Responsables pourront y ajouter des tickets et les contributeurs pourront modifier le devis et le réenvoyer pour validation. Une fois l'itération acceptée par le bénéficiaire, les développeurs doivent produire tous les tickets de cette itération (voir le processus « Gestion des tickets »). La production de tous les tickets de l'itération donne la possibilité au chef d'équipe de démarrer la facturation. Cela est exprimé dans le processus « Gestion des itérations » à travers une contrainte sur le flux de séquence connectant l'action « Accepter itération » et « Facturer itération ». Ce flux est un flux synchrone.

L'action « Facturer itération » du processus « Gestion des itérations » permet de déclencher le processus de facturation « Facturation des facturables » (voir figure 78). Ce déclenchement entraine la création d'une instance « facturationIteration » du processus « Facturation des facturables », que nous avons défini dans le DataStore

du processus, en lui passant le contexte d'exécution du processus parent. L'instance du processus « Facturation des facturables » est un processus imbriqué dans l'instance du processus « Gestion des itérations ». Au déclenchement de l'action « Facturer itération », le contributeur doit saisir un nombre de mois présentant l'échéancier de paiement. Comme illustré à la figure 76, le processus « Facturation des facturables » contient l'attribut « echeanceFacturation » qui correspond à l'échéancier de paiement.

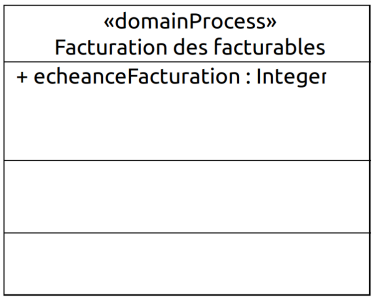


FIGURE 76: Vue informationnelle du processus « Facturation des facturables »

La figure 77 correspond au contexte de réalisation de l'action « Facturer itération ». Celui-ci déclare une InstanceSpecificationFilter spécifiant l'instance « facturationIteration » dans l'état néant ainsi que l'attribut « echeanceFacturation » que le chef d'équipe doit saisir. Le contexte principal est l'instance « iteration » manipulée par le processus « Gestion des itérations ».

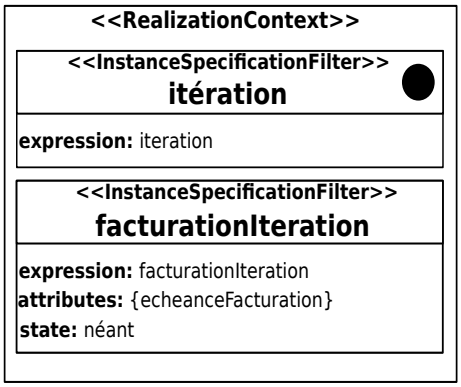


FIGURE 77: Le contexte de réalisation de l'action « Facturer itération »

L'ensemble des gardes, par exemple le fait que tous les tickets doivent avoir une complexité, sont décrites par des expressions OCL (« **iteration.tickets->forAll(t | not t.complexite.oclIsUndefined())** »).

Le processus « Facturation des facturables »

Une fois la facturation lancée, le système se charge de créer une facture dans l'itération selon les données présentes dans le devis. No-

tons que l'itération manipulée par l'instance du processus « Facturation des facturables » est la même que celle manipulée par processus « Gestion des itérations » (c'est-à-dire le processus parent). Le bénéficiaire pourra ensuite payer la facture. Si le paiement est fait dans les temps (c'est-à-dire sans dépasser l'échéance saisie à la création de l'instance du processus « Facturation des facturables ») le système créer une facture d'escompte qui doit être payée par le chef d'équipe au profit du bénéficiaire. Sinon le bénéficiaire paye la totalité de la facture.

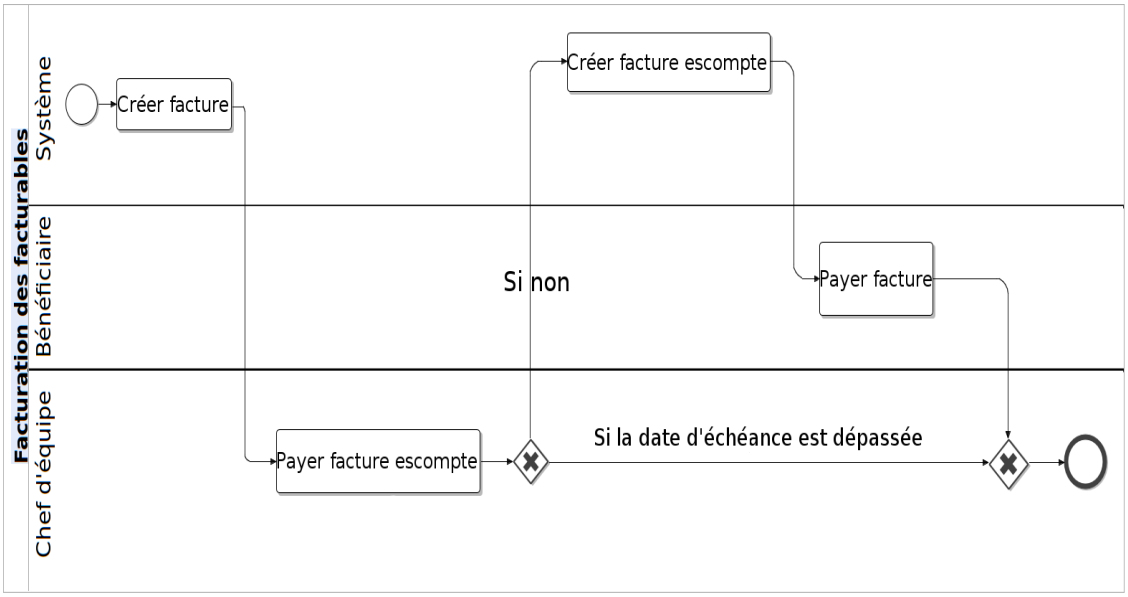


FIGURE 78: Le processus métier « Facturation des facturables »

Afin de vérifier si l'échéance a été dépassée par le bénéficiaire nous avons modélisé un service « ComparateurDeTemps » (voir figure 79) avec la fonctionnalité « echeanceDepassee() » qui prend en paramètre la date de facturation et le nombre de mois (qui représente l'échéancier de l'instance du processus) afin de retourner si l'échéancier a été dépassé par rapport à la date système ou non. La fonctionnalité « echeanceDepassee() » est appelée dans une expression OCL avec la date de création de la facture et l'échéancier du processus comme paramètre.

Le fait que le processus « Facturation des facturables » est un processus imbriqué permet de le réutiliser pour la facturation des « Facturable » d'une manière générale. Par exemple, pour la facturation des formations ou des audits. L'échéance, dans ce cas, est choisie selon le type du facturable.

Le processus « Visualisation de l'espace de travail »

La figure 80 décrit le processus de visualisation des espaces de travail. Ce processus sert à décrire ce que les collaborateurs (Client,

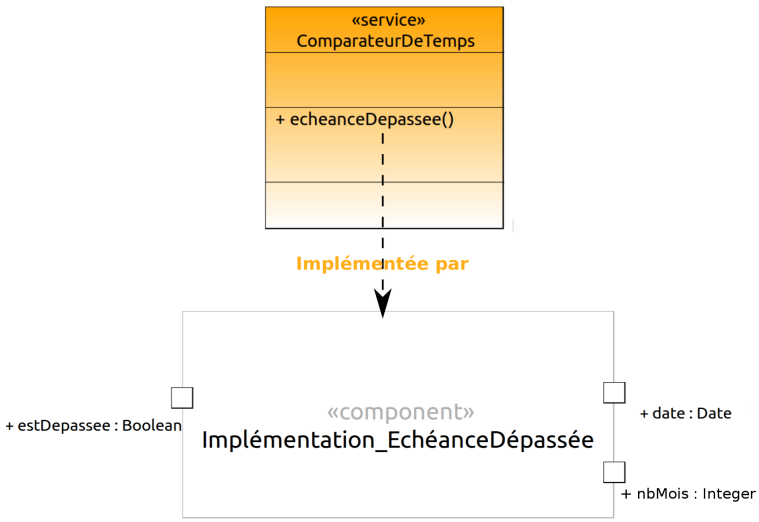


FIGURE 79: Le service de comparaison des dates

développeur, etc.) ont besoin d'étudier, dans le respect des besoins métier, afin de mener à biens leurs rôles dans la collaboration. Les actions décrites dans ce processus sont des actions à déclenchement automatique et permettent de renvoyer une structure de données décrite par le contexte de post-étude. Chaque action a une instance « **eDeTravail** » généralisée de l'objet de collaboration « **EspaceDeTravail** » comme contexte principal. De ce fait, les actions de ce processus sont déclenchables sur une instance d'« **EspaceDeTravail** » quelconque. L'instance « **eDeTravail** » est définie dans le DataStore du processus. Ainsi, nous avons l'action « Voir ET-Client » qui permet de renvoyer, au client connecté, la liste des projets dont il est le bénéficiaire (voir figure 81 a). L'action « Voir ET-Developpeur » qui permet de renvoyer, au développeur connecté, la liste des projets avec, pour chaque projet, son état et la liste des actions qu'il peut effectuer ainsi que la liste des documents et les documents référencés associés à l'espace de travail (voir figure 81 b). Enfin, l'action « Voir ET-Chef de projet » permet de renvoyer, au chef de projet connecté, la liste des factures d'escomptes associées à tous les projets de l'espace de travail avec, pour chaque facture, son état, son montant total et la liste des actions qu'il peut effectuer (voir figure 81 c). Les actions de ce processus sont de type multi instance parallèle. Cela parce que plusieurs collaborateurs peuvent étudier les mêmes instances en parallèle autant de fois que nécessaire.

Sachant que le chef d'équipe est le supérieur des développeurs, nous aurons une concaténation des structures de données renvoyées par les deux actions « Voir ET-Developpeur » et « Voir ET-Chef de projet » pour le chef de projet.

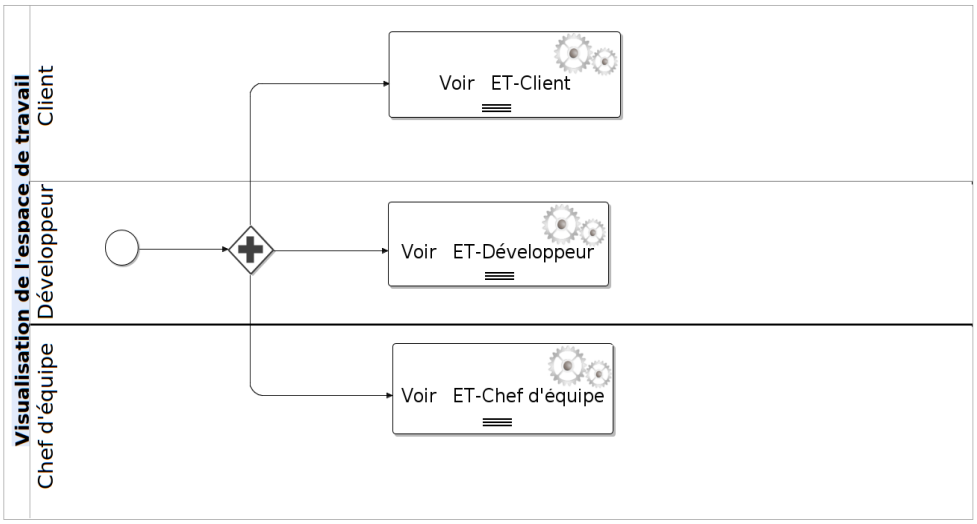


FIGURE 80: Le processus « Visualisation de l'espace de travail »

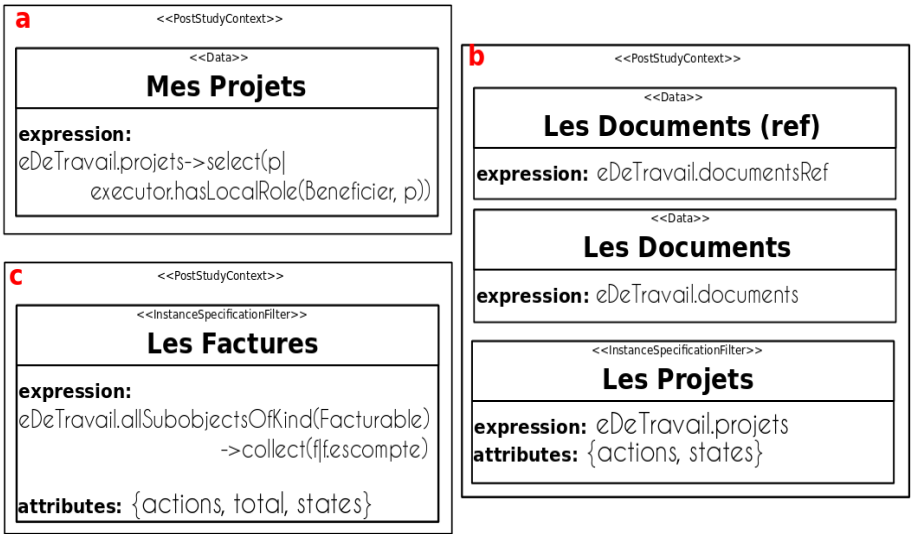


FIGURE 81: Les contextes de post-étude associés aux actions « Voir ET-Client » (a), « Voir ET-Developpeur » (b) et « Voir ET-Chef d'équipe » (c)

8.1.4 *Implémentation à base de composants des actions*

Les actions complexes ou spécifiques doivent être détaillées en fournissant leurs *implementations*. La figure 82 présente celle de l'action « Créer Ticket » du processus « Gestion des tickets » assignée au contributeur. Cette implémentation contient un composant interne, « initialisation du responsable du ticket », permettant d'affecter localement le rôle « ResponsableTicket » au client « c » vers le ticket « t ». Elle contient aussi les données (stéréotypées par le stéréotype *elemntaryData*), le ticket créé, et le client à choisir par le contributeur.

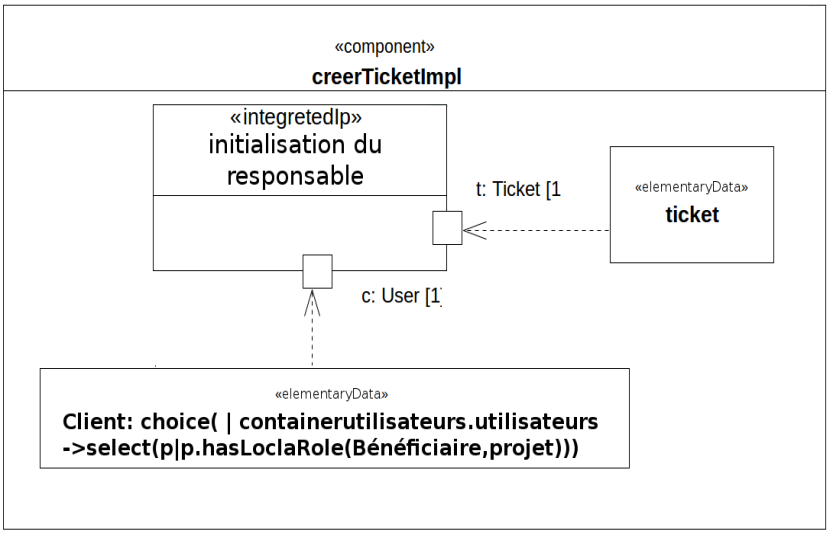


FIGURE 82: L'implémentation de l'action « Créer Ticket »

Le ticket est celui présent dans le data-store du processus. Dans cette implémentation, le contributeur doit pouvoir choisir le client parmi une liste de clients potentiels. Ceci s'exprime par la requête OCL « **choice(| containerutilisateurs.utilisateurs- >select(p | p.hasLocalRole(Bénéficiaire, projet)))** » où le **projet** est le container de l'itération dans laquelle le ticket est créé. L'expression « **choice(| [Liste])** » permet de choisir un élément de la liste proposé en argument. Dans notre cas, la requête OCL permet de récupérer tous les clients présents dans l'instance « containerutilisateurs ».

La figure 83 présente l'implémentation de l'action « Envoyer devis » du processus « Gestion des itérations » assignée au Contributeur. Cette implémentation contient un IP externe, « mailer », permettant d'envoyer un email à une liste de clients, choisie par le contributeur parmi les bénéficiaires du projet contenant l'itération. Ce choix est décrit par l'expression « **choice(| [Liste])** », où la liste est une requête permettant de récupérer les adresses email des bénéficiaires du projet. Le message de l'email est calculé automatiquement à partir des données du devis, le sujet quant à lui doit être saisi par l'utilisateur. Le paramètre « _isUnique » est un paramètre de configuration

qui doit être spécifié au niveau du modèle et permet de choisir le mode d'envoi (envoi unique ou multiple).

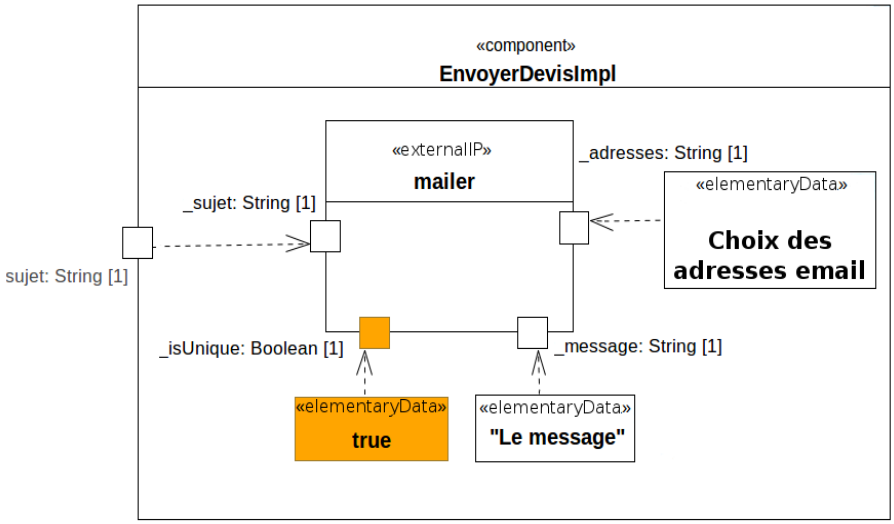


FIGURE 83: L'implémentation de l'action « Envoyer devis »

8.1.5 *L'aspect gestion des réunions*

Certaines parties du modèle peuvent être décrites par des aspects. Par exemple, dans notre modèle le domaine de la gestion des réunions est décrit par l'aspect « Gestion des réunions » (voir figure 84). L'aspect définit des points d'insertion comme « la réunion » sur laquelle l'aspect va s'appliquer, le « profil » avec lequel les réunions auront une relation, ou encore les différentes informations qui peuvent être les « objets de la réunion ». Dans notre cas, l'objet de collaboration « Reunion » correspond à « la réunion » de l'aspect. Les « Facturable », l'« EspaceDeTravail » et les « Projet » sont les « objets de la réunion ». Le profil « Utilisateur » est le « profil » de l'aspect.

L'aspect « Gestion des réunions » va permettre d'ajouter les concepts nécessaires pour la gestion des réunions. Dans notre exemple, seuls les acteurs décrits par le profil Utilisateur et ayant le rôle Développeur auront le droit d'organiser des réunions. Ces réunions seront organisées par rapport aux facturables, aux projets ou aux espaces de travail. Notre objet de collaboration « Reunion » sera complété par la transformation de tissage de l'aspect pour décrire les différentes métadonnées associées aux réunions, par exemple la date de la réunion ou l'ordre du jour, etc. Les détails des processus métier présents dans le modèle du greffon de cet aspect sont présents dans l'annexe B.

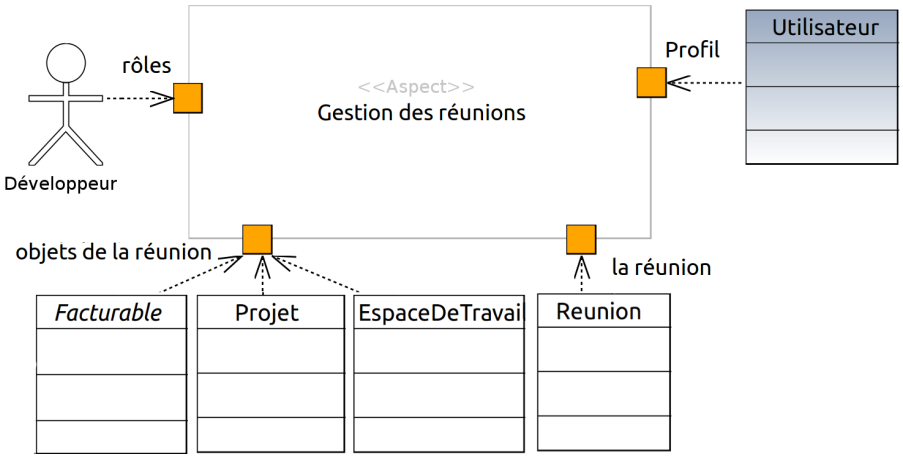


FIGURE 84: L'aspect « GestionDesReunions »

8.1.6 *Génération du portail collaboratif*

Nous avons maintenant assez d'informations pour pouvoir générer le code du portail collaboratif. Le modèle obtenu est introduit comme paramètre d'entrée pour notre processus de production modélisé à l'aide de l'outil **PMS+**. **PMS+** génère une application sous forme de portail collaboratif qui est immédiatement utilisable. Nous avons inscrit dans celle-ci les membres de l'équipe d'Ecréall en leur attribuant des profils (Amen comme développeur, Michael comme chef d'équipe). Nous y avons ajouté le client « bombardier » ainsi que l'espace de travail « Bombardier » dans lequel nous avons créé un projet appelé « SLIC ». Pour rappel (voir section 6.1.5), l'IHM générée par défaut est formée d'un tableau de bord contenant les actions déclenchables, d'un explorateur indiquant l'instance d'objet de collaboration que l'on étudie, et d'un contenu présentant des informations selon l'action déclenchée.

Les captures d'écran suivantes montrent l'évolution du tableau de bord selon le contexte principal (c'est-à-dire l'instance d'objet de collaboration étudiée), la position dans le processus et le rôle de l'utilisateur connecté. La figure 85 montre la racine de l'application qui contient l'instance « userscontainer » décrite dans le modèle, l'espace de travail « Bombardier » ainsi que le container des processus en cours d'exécution « runtime ». Celui-là est une information par défaut décrite dans la bibliothèque de MACoP.

La figure 86 montre les différentes vues obtenues, en fonction des rôles des utilisateurs, à la suite de l'exécution des processus de visualisation (voir figure 80) des espaces de travail. Notons l'héritage de structures de données entre la vue **(b)**, du rôle Développeur, et **(c)**, du rôle Chef d'équipe, décritent dans les contextes de post-étude des actions de visualisation (voir figure 81).

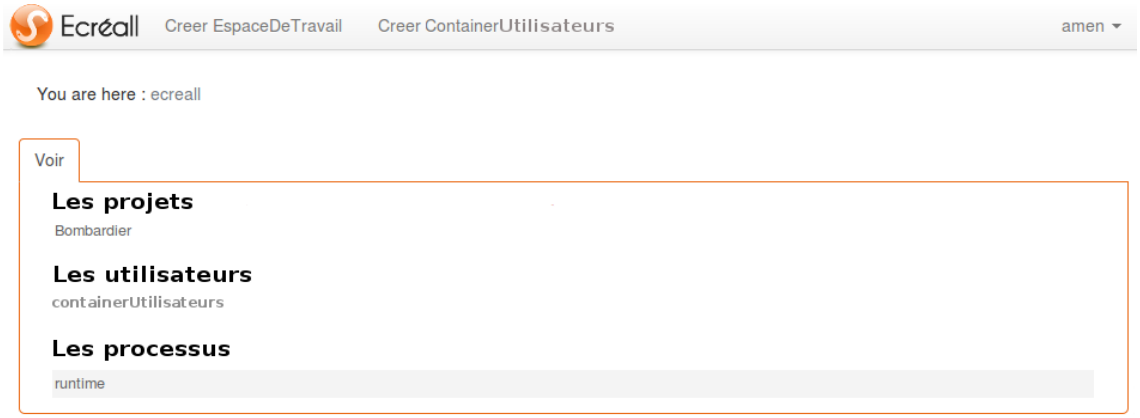


FIGURE 85: Racine de l'application

La figure 87 montre le projet « SLIC » qui contient le « Backlog » ajouté automatiquement à la création du projet et une autre itération « itération1 » ajoutée par le client « bombardier ».

La figure 88 montre le formulaire généré de l'action « Envoyer Devis » de l'« itération1 » exécutée par le Chef d'équipe. Ainsi que le contexte de pré-étude composé par la liste des tickets de l'« itération1 » et les données du devis de celle-ci.

La figure 89 quant à elle montre celui de l'action « Creer Ticket » exécutée par le Contributeur.

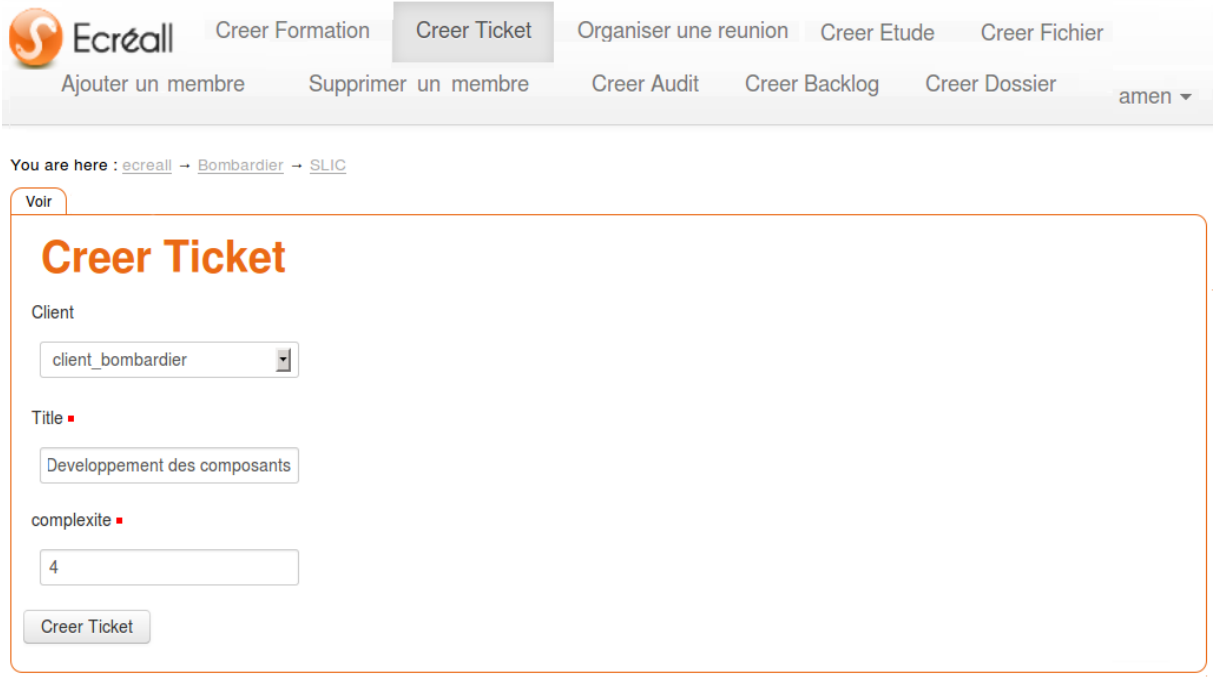


FIGURE 89: Formulaire de l'action « Créer Ticket »

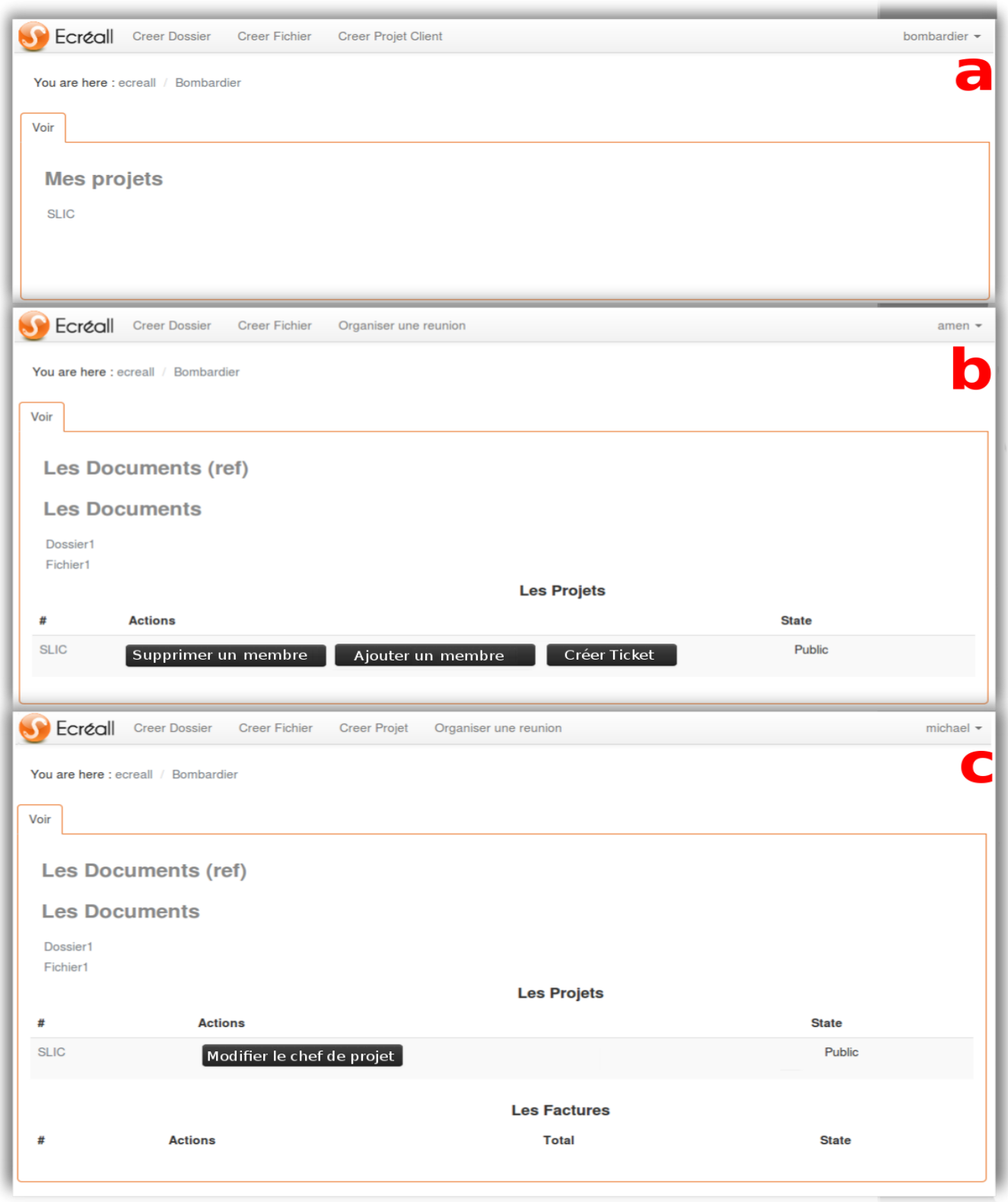


FIGURE 86: Les différentes vues de l'espace de travail « Bombardier »

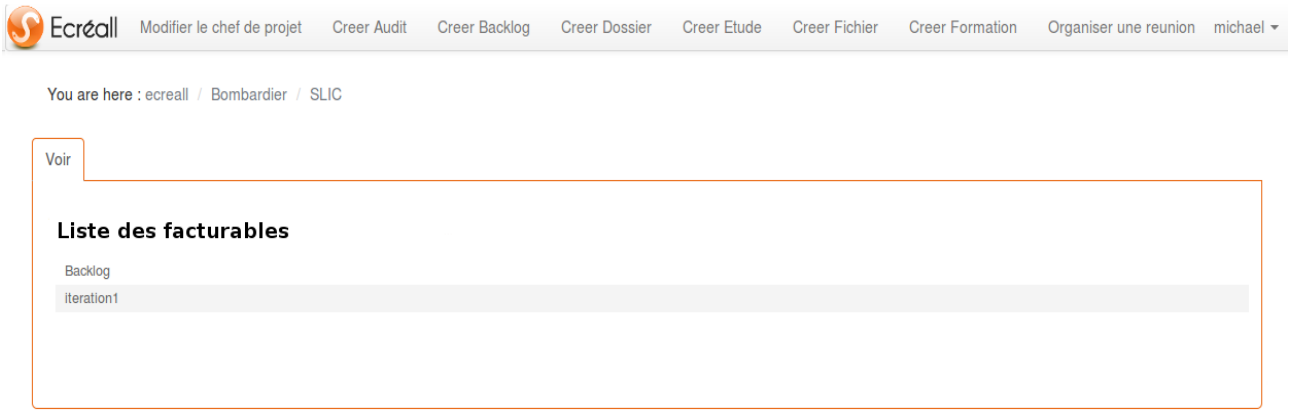


FIGURE 87: Le projet SLIC vue par le chef d'équipe

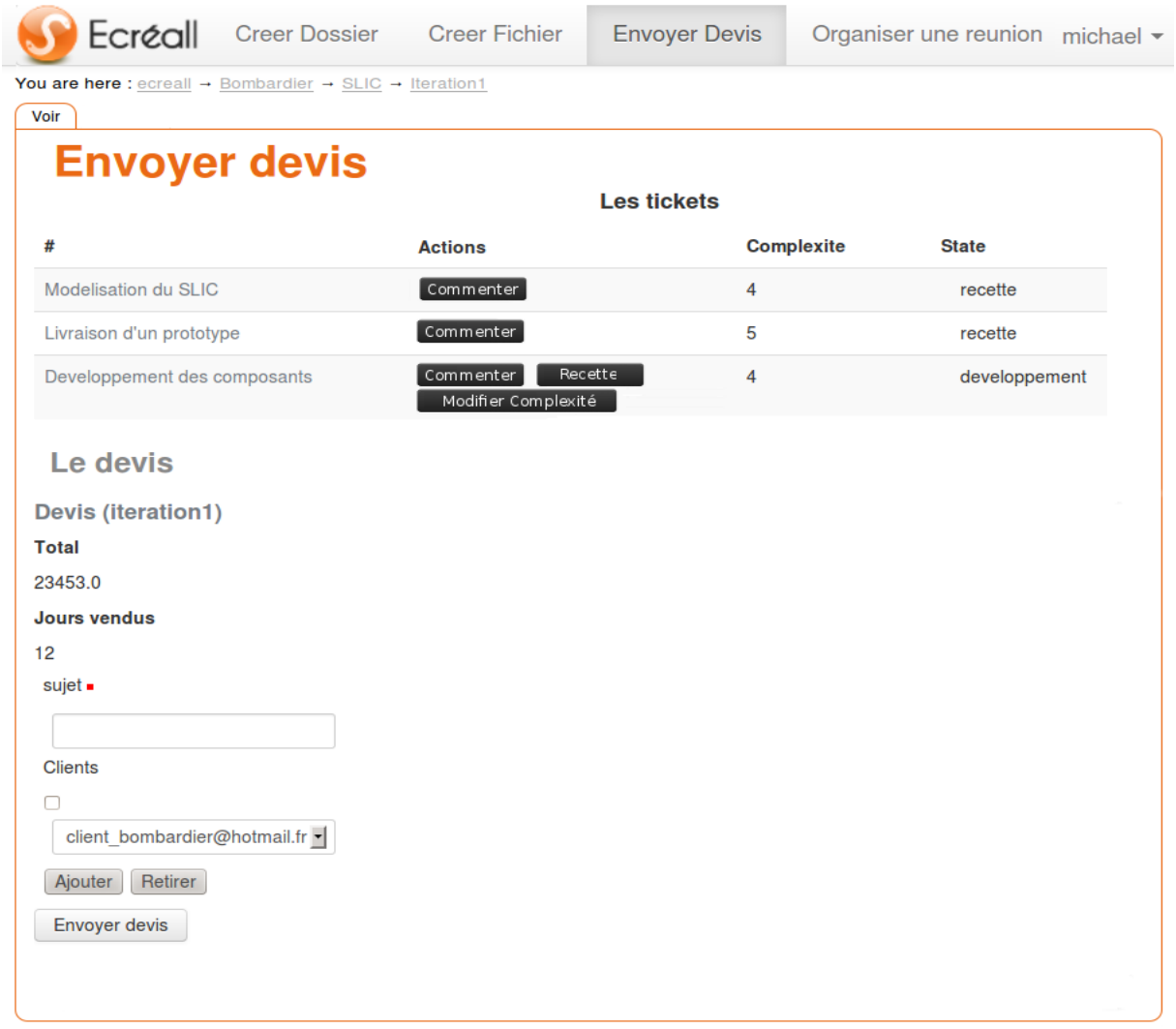


FIGURE 88: Vue de l'action « Envoyer Devis »

8.2 COMPARAISON DE NOTRE MÉTAMODÈLE

Dans le chapitre 5 nous avons discuté de l'inefficacité des approches de modélisation des applications web existantes en raison de leur manque d'accessibilité ou d'expressivité. Dans MACoP nous avons essayé d'établir un équilibre entre l'accessibilité et la complétude afin d'augmenter l'efficacité de l'ergonomie métier. Par rapport aux outils identifiés dans la section 4.3.1 qui adoptent une vision informatique de modélisation (AVI), nous nous sommes intéressés à l'accessibilité en remontant le niveau d'abstraction par une modélisation centrée sur la modélisation des entreprises centrée sur les processus métier. Nous nous sommes intéressés aussi à une séparation des préoccupations sur tous les aspects d'un portail collaboratif. En effet, dans notre approche, le noyau fonctionnel et l'IHM sont totalement séparés. En outre, les différents composants du noyau fonctionnel (c'est-à-dire la vue informationnelle, la vue des ressources et la vue fonctionnelle décrites par la norme ENV 40003) sont séparés les uns des autres. Cela permet d'avoir une modélisation explicite du métier de l'utilisateur, ce qui donne aux acteurs métier un rôle important dans le développement, la maintenance et l'évolution de leur portail collaboratif.

En ce qui concerne l'expressivité, nous nous sommes intéressés à l'ergonomie métier de l'application générée. Cette ergonomie concerne la présence des données de l'application aux utilisateurs en respectant les différentes contraintes et exigences métier (rôles, position dans les processus métier, contexte principal, etc.). Comme décrit dans la section 6.2.2, notre modélisation décrit d'une manière générale l'enchainement des flux de données contrôlé par les actions métier. Pour cela nous utilisons la notion de contexte d'action qui nous permet de capter les différentes informations à montrer à l'utilisateur. Nous l'avons vu dans la section 6.4.1, chaque combinaison entre le type de l'action (automatique ou manuelle) et les contextes de l'action (contexte de réalisation et contexte d'étude) produit un résultat différent en respectant un besoin métier donné. Ainsi, MACoP décrit l'ergonomie métier d'une manière accessible aux acteurs métier.

Dans MACoP l'expressivité ne se limite pas seulement à la présence des données. En effet, afin d'augmenter l'expressivité des exigences métiers, nous avons rendu possible la manipulation des processus métier en tant qu'objet de collaboration. Ainsi, nous pouvons exprimer l'imbrication des processus (c'est-à-dire la modélisation des processus imbriqués à plusieurs niveaux), ce qui permet d'augmenter considérablement l'expressivité de notre modèle. De plus, cela rend possible la modélisation des processus de contrôle et de gestion, d'une manière uniforme et accessible, ce qui n'est pas le cas dans les approches existantes. Nous pouvons aussi citer l'utilisation des implémentations à base de composants réutilisables et adaptables au niveau du modèle ou la réutilisation des modèles avec la modélisation par

aspect ainsi que l'utilisation du langage OCL comme langage de requêtes pour la récupération des données de l'application et les différentes contraintes, etc.

Dans MACoP, nous avons fait le choix de modéliser les processus métier centrés sur les tâches tout en ayant une description des cycles de vie des objets de collaboration. Nous avons aussi étendu le modèle d'exécution BPMN afin de supporter l'exécution centrée sur les données considérée comme plus naturelle. Cette extension nous a permis d'introduire de nouveaux concepts comme nœud de branchement exclusif à choix manuel comme décrit dans la section 6.5. Ces concepts n'ont pas d'équivalent dans les approches existantes, par exemple WebML étant donné que celui-ci n'a pas d'équivalent pour le nœud de branchement exclusif avec plus de deux flux de séquences sortants [21]. En résumé, le métamodèle MACoP a une expressivité équivalente à celles des AVI mais dispose d'une accessibilité plus élevée.

En ce qui concerne les approches décrites dans la section 4.3.3 qui adoptent une vision métier de modélisation (AVM), le modèle de processus métier et les exigences métier d'une manière générale, sont mis à un niveau secondaire dans la modélisation puisque la majorité du travail est fait sur les modèles intermédiaires qui décrivent l'application à un niveau technique (une vision majoritairement informatique). Selon nous, deux problèmes restent à résoudre pour ces approches : le manque d'expressivité au niveau du modèle métier et la génération complète d'une application qui exécute les processus métier de manière implicite et non une simple orchestration de ces processus.

Pour ces approches nous nous sommes intéressés à la complétude en augmentant l'expressivité de notre métamodèle sans affecter son accessibilité. D'une manière générale, nous avons une accessibilité équivalente à celle des modèles d'entrée de ces outils avec une expressivité plus élevée.

Le tableau 1 résume la différence entre notre approche et les approches AVI et AVM. Chaque case du tableau permet de comparer 2 approches entre elles. Dans chacune des cases, nous attribuons un tuple sous la forme $\mathbf{(}\{+|-|\backsimeq\},\{+|-|\backsimeq\}\mathbf{)}$. La première valeur correspond à l'accessibilité et la deuxième correspond à l'expressivité. Ainsi, la valeur

↱	**B**
A	$(+,-)$

Signifie que **A** est plus accessible et moins expressive que **B**. Le signe $\backsimeq$ signifie une équivalence.

En ce qui concerne les techniques de transformations, notre approche se distingue par les points suivants : **1)** Les modifications ultérieures se font uniquement sur le modèle de haut niveau de départ. Aucune modification des modèles intermédiaires n'est nécessaire, comme c'est le cas sur certaines approches ; **2)** Le dernier modèle généré, celui de plus bas niveau, est un modèle représentant la technologie cible. Dans d'autres approches, la génération de code est

↱	**MACoP**	**AVI**	**AVM**
MACoP		$(+, \backsimeq)$	$(\backsimeq, +)$
AVI	$(-, \backsimeq)$		$(-,+)$
AVM	$(\backsimeq, -)$	$(+, -)$	

TABLE 1: Comparaison de MACoP avec les différentes approches.

faite par patron (template) à partir d'un modèle technique, et non technologique, ce qui réduit le niveau de maîtrise et l'optimalité du code produit ; **3)** Le modèle de haut niveau peut être « incomplet », c'est-à-dire que certaines informations implicites peuvent être omises. Elles seront ajoutées automatiquement lors des premières transformations.

8.3 CONCLUSION

Dans ce chapitre nous avons expliqué l'application de notre approche sur une partie du métier d'Ecréall. Dans cet exemple, nous nous sommes intéressés à certains concepts comme l'imbrication des processus, la modélisation des contextes d'actions pour la visualisation des données ou la réutilisation des modèles pour la simplification de la modélisation. L'exemple du métier d'Ecréall nous a permis d'avoir une validation de notre approche au niveau du noyau fonctionnel. D'autres travaux plus ambitieux sont en cours. En effet, dans le cadre d'une collaboration visant à mettre en œuvre un portail collaboratif permettant de produire le programme d'action d'une organisation ou d'un collectif de façon collaborative et démocratique, nous avons modélisé et généré un prototype fonctionnel. Ce projet porte le nom de KuneAgi dont les détails se trouvent sur le site `http://www.kuneagi.org/francais`. Le portail collaboratif généré peut être consulté sur l'adresse suivante `http://testkuneagi.ecreall.com/kuneAgi`.

Ensuite nous avons comparé notre approche à l'existant et nous avons conclu que les principales innovations de notre approche à l'égard des méthodologies actuelles se situent sur trois niveaux : **1)** la génération complète de code à partir d'une modélisation centrée sur le noyau fonctionnel qui décrit le système d'information collaboratif sur ses différentes vues, ce qui permet d'augmenter l'implication des acteurs métier dans la conception des portails collaboratifs (remontée d'abstraction, forte séparation des préoccupations et élargissement de l'espace d'expression) ; **2)** Une technique de transformation 100% automatisée qui exploite des connaissances existantes allant du modèle métier au modèle technologique à grain fin ; **3)** Un portail collaboratif sur mesure, généré, et permettant d'exécuter de manière implicite les

processus métier (c'est-à-dire que les processus métier sont exécutés en arrière-plan). Cette exécution est centrée sur les données ce qui permet à l'utilisateur final de vivre ces processus d'une manière naturelle.

Les travaux en cours concernent l'amélioration et l'optimisation de la génération de code. Étant donnée la démarche centrée sur le noyau fonctionnel que nous adoptons, l'IHM a une forme différente de celle présentée par les autres approches. Pour l'instant, nous considérons l'IHM comme secondaire, mais nous sommes conscients que c'est une partie importante pour l'utilisateur.

Cinquième partie

CONCLUSION ET PERSPECTIVES

9 CONCLUSION ET PERSPECTIVES

Sommaire

Dans ce chapitre, nous commençons par donner un bilan des travaux faits dans cette thèse (section 9.1). Ensuite, dans la section 9.2, nous donnons les différentes ouvertures et améliorations à apporter à notre approche. Certains travaux sont en cours de développement et d'autres sont en cours de validation.

9.1 BILAN

Dans cette thèse, nous nous sommes intéressés à la modélisation et la génération de code complet des portails collaboratifs. Un portail collaboratif est une application web permettant à un ensemble d'acteurs (humains ou machines) de collaborer autour de données selon des processus métier bien définis. Le but de nos travaux est de permettre aux acteurs métier de prendre une grande part dans la modélisation de leurs portails collaboratifs. Notre solution consiste à modéliser les portails collaboratifs à un haut niveau d'abstraction, et ensuite de générer le portail collaboratif entièrement opérationnel. Le travail présenté dans cette thèse est au carrefour de trois domaines de recherche ; la modélisation des entreprises et des systèmes d'information, l'Ingénierie Dirigée par les Modèles (IDM) et la modélisation des applications web.

Dans cette thèse nous avons identifié les différentes approches actuelles de modélisation et de génération des applications web. Nous avons, ensuite, classifié ces approches en deux classes : la première classe adopte une approche centrée sur les aspects techniques (ou informatique) de l'application web (AVI). Cela, selon nous, permet de décrire la solution technique aux besoins métier de l'application.

La deuxième classe quant à elle se focalise sur une modélisation des besoins métier de l'application (AVM). Selon nous, ces deux classes d'approche sont peu efficaces pour que les acteurs métier puissent prendre une part active à la conception de leur application collaborative.

Pour montrer les limites des AVI et des AVM, nous nous sommes basés sur la notion d'ergonomie que nous avons appliquée sur les métamodèles. Nous avons, ainsi, découpé l'ergonomie en deux types : l'ergonomie métier qui se focalise sur la syntaxe concrète ainsi que la sémantique du métamodèle et l'ergonomie du style qui se focalise sur la syntaxe abstraite ainsi que l'interaction offerte par l'outil de modélisation pour la construction du métamodèle. Dans cette thèse nous nous sommes intéressés à l'ergonomie métier. Selon nous, l'ergonomie du style doit être adaptable aux contextes SPPI.

De l'ergonomie métier nous avons déduit un ensemble de critères qu'un métamodèle doit respecter afin d'être efficace. Selon nous, un métamodèle efficace est un métamodèle accessible et complet. Nous avons vu que l'accessibilité est favorisée par la remontée d'abstraction ainsi que la séparation des préoccupations. La complétude quant à elle est favorisée par l'expressivité du métamodèle. De notre analyse des approches existantes de modélisation des applications web, il en découle que les AVI ont une expressivité élevée, mais une accessibilité réduite, ainsi elles sont peu efficaces par rapport aux acteurs métier. Pour les AVM, nous avons une accessibilité élevée, mais une expressivité réduite ainsi elles sont, aussi, peu efficaces par rapport aux acteurs métier. Dans cette thèse nous avons tenté de mettre en place un métamodèle avec une accessibilité et une expressivité élevées.

Notre approche est fondée sur notre métamodèle MACoP accompagné d'un processus de production de logiciels nous permettant de passer d'un modèle MACoP à nos portails collaboratifs en Python. Le métamodèle MACoP permet de décrire les portails collaboratifs à un haut niveau d'abstraction, dans le but de générer le code en suivant une approche IDM. Dans ce métamodèle, nous nous sommes focalisés sur les concepts issus de la modélisation des entreprises centrée sur les processus métier. MACoP présente une rupture par rapport aux approches classiques de modélisation des applications web (AVI). Dans ces approches la modélisation d'une application web est centrée sur la description des pages web et les liens de navigation (c'est-à-dire le système hypertexte) à travers le modèle de navigation. Selon nous, cette modélisation décrit la solution aux besoins métier sous forme d'une architecture visuelle et non les besoins métier.

Dans cette thèse, nous avons commencé par positionner nos travaux par rapport au domaine du génie industriel en proposant la définition du système d'information la plus adéquate à notre démarche. Cette définition sépare le système d'information du systéme physique de l'entreprise. Dans notre définition la machine est un acteur jouant

un rôle dans une collaboration au même titre que l'humain. Pour la spécification de notre métamodèle, nous nous sommes inspirés des travaux faits dans le domaine de la modélisation des entreprises et plus spécifiquement des différents métamodèles comme CIM-OSA et UEML. Ces métamodèles permettent principalement de répondre à la question « Comment » le « Qui » interagie avec le « Quoi » ?. Nous nous sommes, aussi, intéressés à la notion de point de vue de modélisation évoquée par la norme ENV 40003. Cela nous a permis d'organiser notre métamodèle selon ces points de vue.

Notre approche nous a permis de répondre aux trois questions que nous considérons comme fondamentales concernant le fond de l'application, le « Quoi ? », le « Qui ? » et le « Comment ? », afin de constituer un cahier des charges complet sans rentrer dans la complexité des aspects techniques et technologiques du web. La réponse au « Quoi ? » est présent, dans MACoP, comme une modélisation de tout ce qui peut être manipulé dans la collaboration (vue informationnelle d'un système d'information). La réponse au « Qui ? » est présente comme une modélisation des rôles et leur hiérarchie dans la collaboration (vue des ressources d'un système d'information). Enfin, la réponse au « Comment ? » est présentée comme une modélisation des processus métiers (conformes aux métamodèle BPMN) permettant d'établir les relations entre les rôles et les données de l'application (vue fonctionnelle d'un système d'information). Les réponses aux autres questions comme « Où ? » ou « Combien ? » sont considérées comme faisant partie du « Comment ? ».

Dans notre approche le portail collaboratif est vu comme un graphe d'instances des objets de collaborations. Sur chaque instance nous pouvons avoir plusieurs vues. Une vue est un ensemble de données visuelles. Dans nos portails collaboratifs, un utilisateur se déplace d'une instance d'objet de collaboration à une autre. Selon l'instance étudiée par l'utilisateur ainsi que les différentes contraintes (rôle, positions dans les processus métier, etc.), l'application montre un ensemble de vues, qui composent la page web, à l'utilisateur. Dans MACoP, les liens permettant cette navigation sont considérés comme des actions des processus métier. Afin de modéliser les différentes vues, nous nous basons sur ce que nous appelons le contexte d'action. Ce contexte est composé par un contexte de réalisation et un contexte de production, définissant sur quoi l'action s'applique et ce qu'elle produit, ainsi qu'un contexte d'étude définissant les données auxquelles l'utilisateur a accès. Une action, dans MACoP, peut être manuelle, dans ce cas l'action est déclenchée par l'utilisateur de l'application, ou une action à déclenchement automatique, dans ce cas l'action est déclenchée automatiquement par l'application dès que toutes les conditions sont réunies (rôle de l'utilisateur, position dans le processus, etc.). Les choix de modélisation par rapport au type de l'action (à déclenchement manuel ou à déclenchement automatique) ainsi qu'aux

présences ou non de différents contextes de l'action (contexte de réalisation, contextes d'études, etc.) permettent de décrire des comportements différents. Cette technique de modélisation nous permet de nous abstraire par rapport à la notion de liens de navigation, ce qui constitue l'innovation majeure de notre approche.

Notre approche consiste à produire des portails collaboratifs permettant une exécution des processus métier centrée sur les données. Sachant que BPMN permet la modélisation des processus métier centrée sur les tâches, nous avons dû l'étendre afin de supporter l'exécution des processus centrée sur les données. Ainsi nous avons ajouté des concepts comme le nœud de branchement exclusif à choix manuel permettant à l'utilisateur de contrôler manuellement le flux d'exécution des processus métier. Dans nos portails collaboratifs, les processus sont exécutés en arrière-plan. Les utilisateurs agissent sur les données de l'application selon les actions des processus métier.

Par rapport aux AVI, nous avons réussi à avoir une séparation des préoccupations sur tous les niveaux. D'abord nous avons une séparation entre le noyau fonctionnel et l'IHM. En effet, nous ne raisonnons pas en terme de modèle de navigation, mais nous raisonnons en terme de données à étudier. Dans notre approche les préoccupations du NF, elles-mêmes, sont séparées selon la norme ENV 4003 (la vue fonctionnelle, la vue informationnelle et la vue des ressources). De plus, le fait de se baser sur la modélisation des entreprises centrée sur les processus métier nous permet d'avoir une modélisation à haut niveau d'abstraction ce qui situe notre approche à un niveau plus abstrait que les AVI.

Par rapport aux AVM, nous avons réussi à élargir l'espace d'expression de notre métamodèle. Cela en se basant sur des techniques de modélisation comme l'imbrication des processus métier (processus imbriqué) ou l'utilisation du langage OCL à tous les niveaux (contraintes, description des données, etc.). L'utilisation des contextes des actions ainsi que l'extension du modèle d'exécution de BPMN nous a permis de ne pas nous limiter aux applications d'orchestration de processus métier et de modéliser des applications centrées sur les données. Nos portails collaboratifs sont donc des applications web centrées sur les données et sur les processus métier.

Afin de simplifier la modélisation et permettre la réutilisation de modèle, notre approche propose la modélisation orientée aspect. Avec cette modélisation le concepteur peut modéliser son application par composition de modèle. Cela permet de mettre en place rapidement un portail collaboratif opérationnel.

Enfin, pour valider notre approche, un premier processus de production de logiciels vers la technologie Python a été développé. Ce processus de production nous permet de générer 100% du code fonctionnel pour les portails collaboratifs permettant d'exécuter les processus métier de la collaboration d'une manière implicite et transpar-

ente. Notre processus de production est modélisé et exécuté par notre outil PMS+ que nous avons développé afin de résoudre certains problèmes dans les outils existants d'enchainement de transformations dans une démarche IDM, par exemple la dépendance aux technologies utilisées par les transformations.

Le métamodèle MACoP est totalement indépendant des technologies cibles des portails collaboratifs. Il est, donc, possible de développer d'autres processus de production pour d'autres technologies, par exemple J2EE, PHP, etc. Ces processus de production peuvent réutiliser une partie du processus de production actuel, notamment une partie des transformations de modèles et des métamodèle intermédiaires de haut niveau.

Nous avons appliqué notre processus de production de logiciels sur un modèle MACoP décrivant le métier d'Ecréall. Cela a permis de générer un premier portail collaboratif permettant le suivi des projets de développement d'Ecréall. Un projet de grande envergure est en cours, nous avons réalisé un prototype fonctionnel permettant de produire le programme d'action d'une organisation ou d'un collectif de façon collaborative et démocratique. Ce projet porte le nom de KuneAgi dont les détails sont sur le site du projet `http://www.kuneagi.org/francais`. Le portail collaboratif généré peut être consulté sur l'adresse suivante `http://testkuneagi.ecreall.com/kuneAgi`. Ce projet constitue un premier pas vers l'industrialisation de notre approche.

Depuis 2005 Ecréall utilise une modélisation partielle du métier de ses clients pour générer les composants permettant d'adapter le gestionnaire de contenu Plone [71] au besoin de ses clients, en terme de portails collaboratifs, grâce à ArchGenXML [110]. Les clients demandent fréquemment à Ecréall de modifier les modèles et le code pour adapter le portail à leurs nouveaux besoins, nos travaux ouvrent une nouvelle voie, non seulement la totalité du code est générée, mais avec un DSL adapté au métier du client il devient possible de laisser celui-ci modifier lui même son portail.

En conclusion, nous pouvons dire que notre approche ouvre une nouvelle voie qui s'avère prometteuse pour une utilisation plus large des outils de modélisation et de génération automatique complète des portails collaboratifs. Nous pouvons dire aussi, à travers cette approche, que nous atteignons la finalité de ces travaux qui est de donner à l'entreprise la possibilité de faire évoluer son portail collaboratif lors des changements de son métier, et ce, en toute autonomie. Le metamodèle et l'ensemble du processus de production sont disponibles sous licence AGPL (`https://gforge.inria.fr/projects/omegsi/`).

9.2 PERSPECTIVES

Dans l'état actuel de nos recherches, de nombreux travaux restent à faire allant de la modélisation à la génération de code. Dans ce qui suit, nous exposons ces travaux.

9.2.1 *Génération de l'IHM*

Actuellement nous générons une interface utilisateur basique déduite du modèle des exigences métier. Cette IHM offre à l'utilisateur de l'application le minimum nécessaire respectant son métier. Les exigences des utilisateurs ne se limitent pas seulement au noyau fonctionnel, mais aussi à l'IHM. En effet, les plateformes actuelles, avec leur puissance de calcul, offrent une panoplie de possibilités en terme d'interaction homme-machine. Ainsi, la représentation des données n'est plus limitée à deux dimensions. On trouve, actuellement des applications à quatre dimensions, où la quatrième dimension présente le temps (pour les applications web temps réel, par exemple). L'interaction quant à elle a évolué pour devenir de plus en plus naturelle (commandes vocales ou tactiles par exemple). Il existe aujourd'hui des méthodologies permettant la génération des IHMs à partir d'une description à haut niveau d'abstraction. Cette description est souvent fusionnée avec la description du noyau fonctionnel. Étant donné l'approche orientée par le noyau fonctionnel que nous adoptons, la description de l'IHM aura une forme différente. Pour cela, nous souhaitons créer un métamodèle des IHM et un métamodèle des contraintes ergonomiques (ergonomie du style) qui associés au modèle métier par la stratégie de transformations permettront de produire les interfaces hommes machines conviviales et ergonomiques, par exemple faire en sorte que l'accès à une fonctionnalité doit être à une distance courte de celle en cours, c'est-à-dire, à moins d'un clic ou d'un raccourci clavier (la notion de distance). De même, nous voudrions à terme créer un modèle ontologique de l'application, qui associé au modèle ergonomique permettra par exemple de demander confirmation pour certaines actions, comme lors d'une suppression des contenus. Nous voulons également pouvoir enrichir automatiquement les modèles à partir de statistiques des actions des utilisateurs, on pourrait alors tenir compte du fait que la fonction *undo* est systématiquement utilisée après telle action et modifier les modèles pour demander confirmation ou afficher une prévisualisation du résultat.

9.2.2 *Modélisation des aspects synchrones*

Le portail collaboratif généré par notre outil permet l'affichage des données visuelles sans aspect synchrone. La synchronisation se traduit par des micros communications (échange des données) en-

tre le client et le serveur. Ces communications peuvent entrainer le changement d'une partie ou la totalité d'une page web. Elles sont, aussi d'une manière générale, faites automatiquement. Les aspects synchrones d'une application web sont la base des applications internet riche. Dans MACoP, la modélisation des aspects synchrones à un haut niveau d'abstraction n'est pas encore étudiée. Par exemple, dans une application de type messagerie instantanée nous trouvons les messages (messages échangés) ainsi que les utilisateurs (qui échangent les messages) comme objets de collaboration. Dans cette application, nous trouvons l'initiateur de la discussion et les participants à la discussion comme rôles. L'initiateur est le créateur d'une discussion. Ce rôle donne, à un utilisateur, le droit de rendre la discussion publique ou privée ainsi qu'inviter des utilisateurs à la discussion, etc. Les participants (utilisateur invité à la discussion) quant à eux ont le droit de poster des messages. Chaque participant voit l'ensemble des messages échangés en temps réel. La structure des données, formée par l'ensemble des messages vus par les participants, dans ce cas, doit être synchrone. Ce qui n'est pas possible dans MACoP actuellement.

Nous trouvons aussi les aspects synchrones dans les valeurs des champs des formulaires. Par exemple, sur un site de vente de voitures d'occasion, un client est amené à faire certains choix. Il commence par choisir la marque de la voiture, le champ suivant lui propose les modèles de la marque choisie dans le premier champ. Les champs des formulaires peuvent dépendre les uns des autres. Ces dépendances peuvent être cycliques ou non. Les différents champs d'un formulaire, dans ce cas, peuvent être décrits par un graphe cyclique. Dans notre approche, les formulaires sont générés à partir du contexte de réalisation de l'action, ainsi qu'à partir des paramètres des composants décrivant l'implémentation de l'action. Nous pensons qu'une description des dépendances entre ces paramètres suffit pour décrire les aspects synchrones entre les champs des formulaires générés.

Les outils actuels permettent la modélisation des aspects synchrones d'une application web à un niveau technique. La modélisation concerne la solution technique et non le besoin métier qui a conduit à cette solution. Étant donné l'approche centrée sur le noyau fonctionnel que nous adoptons, nous souhaitons ajouter à MACoP la possibilité de traiter les aspects synchrones, à un niveau métier, d'une application afin de voir l'évolution des structures des données en temps réel par exemple (nous parlons ici des applications web riches). Ces travaux sont en cours.

9.2.3 *Modélisation des processus d'implémentation pour la génération des formulaires à structure dynamiques*

Dans notre modélisation les actions ont une implémentation à base de composants. Dans ce cas, les composants sont connectés les uns aux autres par leurs paramètres. Cela permet de déduire l'enchainement d'exécution des composants. Cet enchainement ne constitue pas un flux d'exécution conditionnel. Dans ce cas, tous les composants doivent être exécutés au déclenchement de l'action. Mais, ce déclenchement simultané de tous les composants n'est pas toujours désirable. En effet, certaines actions doivent être faites par étapes. Par exemple, dans un processus de facturation, l'action de paiement peut être faite en plusieurs étapes. La première étape permet au client de choisir son mode de paiement, la deuxième lui demande son adresse de livraison et la finale lui demande de saisir les données utiles au paiement selon le mode de paiement choisi dans la première étape. L'enchainement de ces étapes constitue un processus d'implémentation conditionnel. Ces processus ne durent pas dans le temps et décrivent l'implémentation d'une action métier. À partir des processus d'implémentation, nous voulons générer des formulaires à structure dynamique permettant de saisir les données de l'action sur plusieurs étapes.

Le découpage d'une action, au niveau de son implémentation, en plusieurs étapes sous forme de processus d'implémentation ne peut être fait par processus métier, par exemple par un sous-processus. En effet, le processus métier permet de modéliser un enchainement d'action qui s'étale sur le temps. Dans ce cas, les actions peuvent être exécutées par l'utilisateur séparément à tout moment. Dans les processus d'implémentation, l'enchainement des étapes est fait automatiquement par l'application et constitue une seule exécution. Cela signifie que toutes les étapes doivent être faites sur un même laps de temps (c'est-à-dire durant une session). Si l'utilisateur abandonne l'exécution de la troisième étape par exemple, c'est toute l'action qui n'est pas exécutée. L'utilisateur doit, dans ce cas, exécuter à nouveau toutes les étapes. Ce qui n'est pas le cas si la modélisation est faite par les processus métier.

Afin de réaliser cela, nous souhaitons remplacer le modèle de composants par un modèle de processus d'implémentation (par exemple, par une fusion entre le diagramme de composants et les processus métier BPMN). Cela donnera la possibilité de générer des formulaires à structure dynamique avec des dépendances entre les champs des formulaires. Ces travaux sont en cours de validation.

9.2.4 *Réalisation parallèle des actions*

Dans une collaboration, les acteurs peuvent effectuer certaines actions en parallèle, par exemple rédiger une facture ainsi qu'un rapport d'activité. Ces actions appartiennent, généralement, à des processus métier différents. Dans nos portails collaboratifs, les actions sont réalisées (voir le processus de réalisation d'une action dans la section 6.4) d'une manière séquentielle. En effet, dans nos portails collaboratifs, une fois qu'une action est déclenchée par un utilisateur, l'application montre le formulaire associé. Si l'utilisateur déclenche une autre action, le premier formulaire est supprimé et le deuxième est affiché. Nous disons, dans ce cas, que les réalisations des actions sont séquentielles. Dans le cas des réalisations parallèles des actions, le deuxième formulaire est affiché avec le premier formulaire sur une même page. L'utilisateur pourra valider un formulaire puis valider le deuxième. Nous pensons que les relations (parallèle, séquentiel) qu'une action peut avoir avec les autres actions tiennent des pratiques de travail des acteurs au sein d'une collaboration. Nous voulons décrire, dans MACoP ces relations, et générer le code correspondant.

9.2.5 *L'implémentation du passage du niveau informel au niveau formel*

Comme décrit dans la section 3.3, l'IDM permet de couvrir la phase de modélisation des exigences métier. Nous identifions cette phase comme le passage du niveau informel au niveau formel des besoins métier. Ce passage demande des capacités d'analyse et d'interaction élevées. Le modèle source étant informel, il peut être sous forme textuelle (description des scénarios en français par exemple). Nous souhaitons avoir un outil permettant d'interagir avec l'utilisateur afin de lever l'ambiguïté de son modèle et de l'enrichir. Cela afin de construire un modèle formel de ses besoins métier.

9.2.6 *Génération vers d'autres technologies*

Sachant que MACoP est totalement indépendant de la technologie cible, à terme nous souhaitons réaliser d'autres stratégies de transformation vers d'autres technologies, par exemple J2EE, PHP, etc. Cela implique, aussi, la nécessité de générer les scripts de migration des données d'une technologie vers une autre. L'utilisateur, dans ce cas, n'aura pas de difficultés à faire évoluer son portail en un temps et coût réduit.

D'autres travaux doivent être réalisés comme le test et la vérification ou la simulation et l'optimisation des processus métier.

Sixième partie

ANNEXES

A

LA NOTATION BPMN

A.1 LES CONCEPTS DE BASE DE BPMN

Concept	Description	Notation
Evènement	Un évènement est un changement d'état du système. Ce changement affecte le flux d'exécution d'un processus métier. Dans BPMN, un évènement peut être un évènement déclencheur, par exemple signaler une erreur, ou un évènement résultant, par exemple la réception d'un message.	
Activité	Une activité est un ensemble d'opérations (travail) à effectuer dans l'entreprise. Une activité peut être élémentaire (action) ou décomposée (sous-processus).	
Branchement (Passerelle)	Un nœud de branchement (ou passerelle) permet de contrôler le flux d'exécution d'un processus métier. Cela en fusionnant ou en divergeant des chemins d'exécution.	
Flux de séquence	Un flux de séquence permet de décrire l'enchainement d'exécution des activités du processus métier	
Piste	Une piste permet de délimiter les activités d'un processus métier. Une piste peut décrire un responsable ou une organisation exécutant ce processus.	
Corridor (*Lane*)	Un corridor (ou *Lane*) permet de regrouper des activités dans un processus métier. Il se présente comme une subdivision d'une piste. Un corridor peut décrire à qui les activités sont affectées.	

A.2 EXEMPLES DE TYPES D'ÉVÈNEMENTS DANS BPMN

Type	Initial	Intermédiaire	final
Evènement quelconque			
Evènement conditionnel			
Evènement timer			

A.3 EXEMPLES DE TYPES DE BRANCHEMENTS DANS BPMN

Type	Description	Notation
Exclusif	Le branchement exclusif est activé dès la réception d'un jeton. À la réception du jeton, il produit un seul jeton sur le premier flux vérifiant la garde. Si aucun flux ne vérifie la garde, le jeton est produit sur le flux par défaut.	
Parallèle	Le branchement parallèle est activé dès la réception de tous les jetons des flux rentrants. À la réception des jetons, il produit un jeton sur chaque flux sortant.	
Inclusif	Le branchement inclusif produit un jeton sur chaque flux sortant vérifiant la garde.	

A.4 AUTRES NOTATIONS BPMN

Concept	Description	Notation
Action multi instance parallèle	L'action est instanciée plusieurs fois. L'exécution de ces instances d'action est faite d'une manière parallèle c'est-à-dire que les utilisateurs, dans ce cas, exécutent parallèlement ces instances.	
Action multi instance séquentielle	L'action est instanciée plusieurs fois. L'exécution de ces instances d'action est faite d'une manière séquentielle c'est-à-dire que les utilisateurs, dans ce cas, exécutent séquentiellement (l'un après l'autre) ces instances.	
Sous-processus	Un sous-processus permet de cacher une partie du processus. Il permet, aussi, de hiérarchiser le processus métier.	

B

EXEMPLE D'ECREALL

B.1 L'ASPECT «GESTION DES RÉUNIONS»

Dans le modèle du greffon de l'aspect « Gestion des réunions » nous avons défini un processus métier, illustré à la figure 90, permettant la gestion de la réunion. Nous avons défini, dans ce processus, un point d'insertion identifié par le point d'interrogation sur la figure. Ce point d'insertion correspond au rôle permettant à son détenteur d'organiser une réunion. Dans l'exemple d'Ecréall (voir section 8.1.5), ce point d'insertion correspond au rôle Développeur. Ainsi, un Développeur commence par organiser une réunion en choisissant les participants et en fixant l'ordre du jour ainsi que les différentes propositions de date de la réunion. Le Développeur devient l'organisateur ce qui lui donne le droit de modifier les données de la réunion, inviter et désinviter des membres, et fixer le président de la réunion. Une fois ce dernier choisi, les processus de vote peuvent être démarrés. L'action « Demarrer le vote » permet de créer une instance de processus d'élection « Election » (voire figure 91) par participant. Ces processus sont personnels et ne peuvent être exécutés que par le participant électeur. En effet, le rôle « Electeur » est un rôle affecté localement par rapport à l'instance de processus. Dans notre cas cette instance est « self » (c'est-à-dire l'instance de processus, elle même). Chaque « Electeur » exécute son processus d'élection indépendamment des autres. Un électeur peut refuser de participer, accepter l'invitation si celle-ci ne propose qu'une seule date, ou voter une ou plusieurs dates dans le cas contraire.

Une fois que l'électeur a voté, il pourra modifier son vote tant que la réunion n'est pas confirmée. La confirmation de la réunion par l'organisateur permet d'envoyer un email à tous les participants qui ont voté pour la date qui a récolté le plus de vote. À la date de la réunion, le président pourra rédiger le compte rendu de la réunion, et le modifier, une fois la réunion cloturée, un mail sera envoyé automatiquement à tous les participants de la réunion.

La figure 92 montre la capture d'écran de l'action « Organiser une reunion » du processus « GestionDesReunions » décrit dans notre modèle par le mécanisme de la modélisation par aspect.

La figure 93 montre celle de l'action « Voter » de l'utilsateur inviter à la réunion.

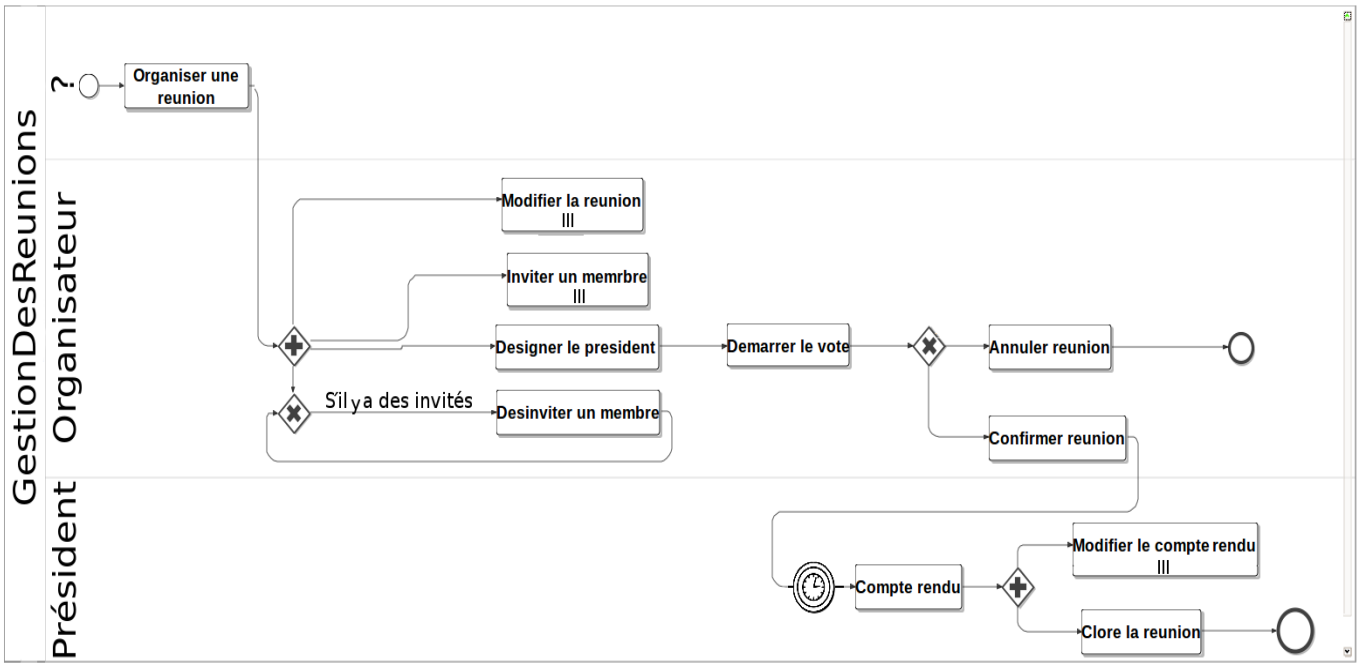


FIGURE 90: Le processus métier « GestionDesReunions »

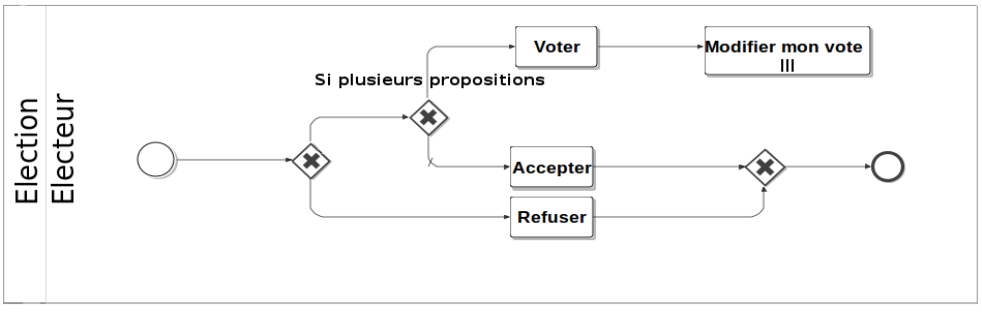


FIGURE 91: Le processus métier « Election »

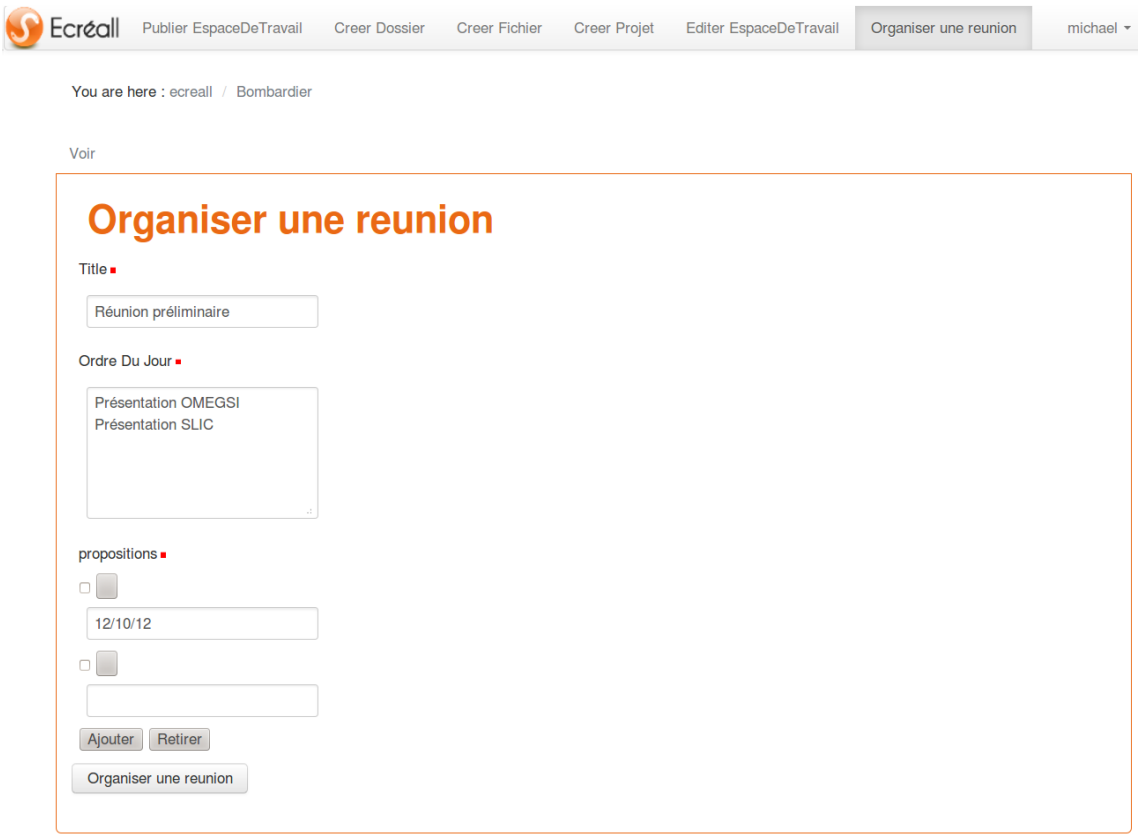


FIGURE 92: Formulaire de l'action « Organiser une reunion »

FIGURE 93: Formulaire de l'action « Voter »

B.2 LE PROCESSUS MÉTIER «GESTION DES PROJETS»

Dans le processus métier « Gestion des projets », illustré à la figure 94, c'est le Client ou le Chef d'équipe qui peuvent initier le processus. Une fois le projet créé le chef d'équipe pourra désigner le chef de projet. Celui-là doit désigner l'équipe qui va travailler sur le projet. Ces derniers deviennent des Contributeurs au projet. Le Bénéficiaire dans ce cas pourra démarrer le projet. Les développeurs contributeurs au projet pourront créer des tickets et les produire, etc.

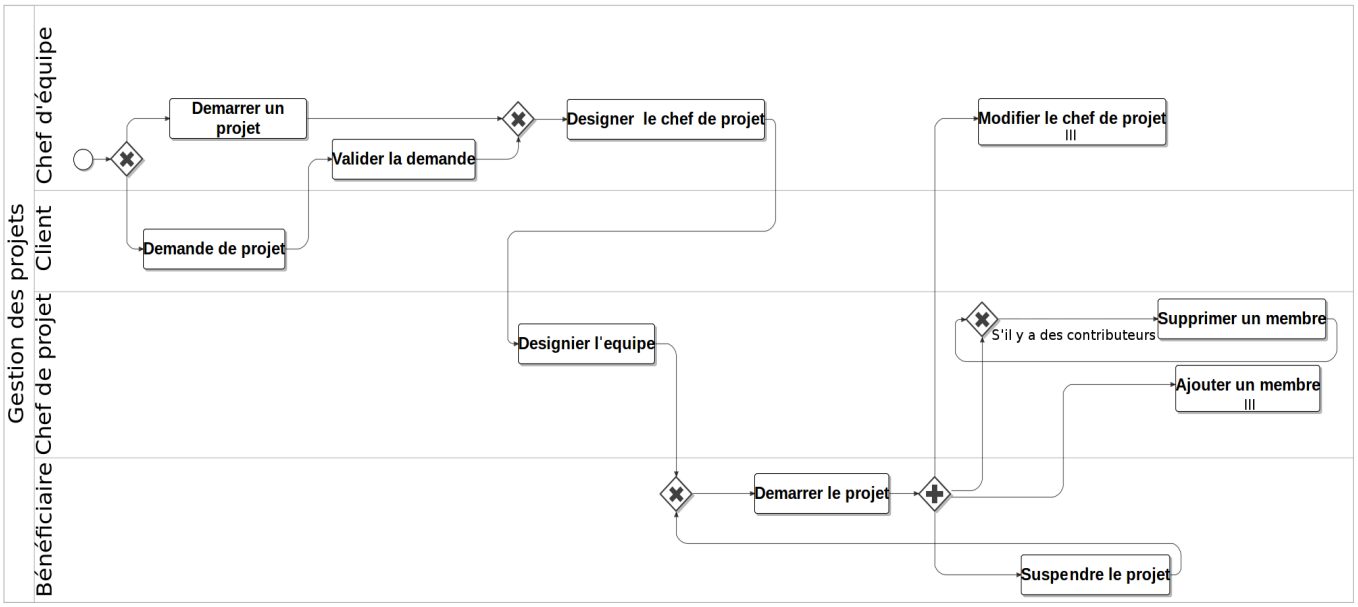


FIGURE 94: Le processus métier « ProjetCreation »

GLOSSAIRE

accessibilité	L'accessibilité d'un métamodèle signifie que le métamodèle est facilement compréhensible par le concepteur, 76, 77, 82, 98, 198
acteurs métier	Une entité active (un humain ou une machine) faisant partie du système physique. Ces entités sont chargées d'exprimer des besoins métier ainsi qu'agir selon des processus métier bien définis, 3, 14, 27, 28, 77, 97, 108, 190, 197
action	Un ensemble d'opérations élémentaires agissant sur un ensemble de données, 15, 26, 27, 87, 98, 105, 114, 120, 128, 199
action à déclenchement automatique	C'est une action déclenchée automatiquement dès que toutes les conditions sont satisfaites. Ce type d'action peut-être exécutée par des humains ou des machines, 125, 182, 199
action automatique	C'est une action exécuter automatiquement par la machine sur laquelle l'application s'exécute, 125
approche fonctionnelle	Dans cette approche, le système d'information est présenté comme un ensemble de fonctions permettant d'assurer la disponibilité de la bonne information pour la prise des décisions., 20
approche structurelle	Dans cette approche, le système d'information est composé de deux sous- systèmes ; le système de traitement d'information et le système informatique. Le système de traitement de l'information décrit comment les acteurs collaborent en agissant sur les informations pour atteindre un objectif de production selon des processus métier bien définis., 21

approche systémique
: L'approche systémique se définit comme une méthodologie permettant de cerner les connaissances par rapport à un système donné à fin de mieux agir sur ce dernier, 20

architecture visuelle
: Une architecture où l'application web est assimilée à un graphe de nœuds interconnectés. Les nœuds représentent les pages web et les arcs représentent les liens de navigation entre ces pages, 62, 78, 132, 198

AVI
: Approche de modélisation des applications web privilégiant une Vision Informatique de l'application, 56, 71, 77, 80–82, 190, 191, 197

AVM
: Approche de modélisation des applications web privilégiant une Vision Métier de l'application, 56, 71, 77, 81, 82, 191, 198

BAM
: Business Activity Monitoring, 31, 127

BPM
: Business Process Management, 31

collaboration
: Travail d'un ensemble d'acteurs selon des processus métier bien définis afin de produire de la valeur ajoutée, 14, 20, 23, 32

complétude
: La complétude d'un métamodèle signifie que le métamodèle doit décrire tous les aspects du système, 75–77, 82, 198

contexte d'étude
: Ce contexte permet de spécifier ce que l'utilisateur de l'application doit étudier avant d'effectuer l'action (contexte de pré-étude) ainsi qu'après le traitement de l'action (contexte de post-étude), 90, 123, 128, 133, 190, 199

contexte d'action
: Ensemble de contextes permettant de spécifier ce que l'action produit (contexte de réalisation), où elle le produit (contexte de production) et ce que l'utilisateur a besoin d'étudier afin de réaliser cette action (contexte d'étude), 115, 121, 123, 124, 128, 132, 147, 190, 199

contexte d'exécution
: Ensemble de données ayant une relation forte (c'est-à-dire identifiable) avec le processus, 126, 127, 180

contexte de production
: Le contexte de production permet de décrire les états des instances des objets de collaboration, après le traitement de l'action, ainsi que leurs nouvelles destinations en cas de changement de celle-ci, 123, 128, 177, 199

contexte de réalisation
: Le contexte de réalisation permet de spécifier sur quoi l'action s'applique (contexte de production), ainsi que l'ensemble des contraintes associées à l'état des instances des objets de collaboration, et également l'ensemble des attributs de ces instances qui sont manipulés directement par l'utilisateur dans le cadre de cette action, 89, 121, 133, 135, 180, 190, 199

contexte principal
: L'instance d'objet de collaboration associé à l'action à exécuter, 98–100, 122, 132, 133, 135, 138, 177, 190

donnée
: C'est une donnée d'une manière générale appartenant au noyau fonctionnel de l'application. Une donnée peut être une instance d'un objet de collaboration, la valeur d'un attribut d'un objet de collaboration ou le résultat d'une expression, 100, 114, 115, 119, 120, 128

donnée visuelle
: Les données visuelles sont les informations présentes sur une page web. Ces données peuvent être des données visuelles statiques, comme un texte ou une image, ou des données visuelles d'interaction, par exemple un champ d'un formulaire, 99, 132, 134, 199

donnée visuelle d'interaction
: Les données visuelles permettant à l'utilisateur d'interagir avec l'application, par exemple des formulaires, 132

donnée visuelle informative
: Les données visuelles permettant d'informer l'utilisateur de l'application, comme un texte ou une image, 132

DSL
: Domaine Specific Language (langage spécifique au domaine), 65, 68, 85, 97, 201

DSTL
: Domaine Specific Transformation Language (langage de transformation spécifique au domaine), 142

EA — Le modèle d'entité-association permet de décrire la structure de données sous la forme d'un diagramme contenant des entités ainsi que des associations entre ces entités, 26, 57, 61

ergonomie du style — L'ergonomie du style d'une application d'une manière générale se préoccupe de la présentation des données et l'interaction entre l'application et l'utilisateur les plus adaptées à la plateforme ainsi qu'aux utilisateurs, 74, 102, 133, 147, 198

ergonomie métier — L'ergonomie métier d'une application d'une manière générale se préoccupe de la présence des données ainsi que des actions au bon moment au bon endroit pour la bonne personne. Cela dans le respect des exigences métier de l'application, 74, 102, 133, 147, 190, 198

espace d'expression — C'est la portée de l'expressivité d'un métamodèle, 75, 78, 81, 200

expert métier — Personne chargée d'identifier les exigences métier d'une application, 43

expressivité — On dit qu'un métamodèle est expressif s'il permet aux concepteurs d'exprimer tous leurs besoins. Le niveau d'expressivité varie d'un métamodèle à un autre. L'expressivité dépend de l'utilité du métamodèle. Dans cette thèse le contexte de l'expressivité est la modélisation des applications web, 72, 75, 81, 82, 85, 98, 147, 198

flux de séquences — Dans un processus métier BPMN, les flux de séquences permettent de connecter les noeuds (actions, événements, noeuds de branchement, etc.). De ce fait, ils décrivent le flux de contrôle de l'exécution des noeuds (c'est-à-dire l'enchainement des noeuds), 116, 118, 137, 178

formalisation — Le passage d'une description métier informelle des besoins métier à une description métier formelle, 40

IDM — Ingénierie Dirigée par les Modèles, 4, 38, 85, 197

IHM	Interactions Homme-Machine, 4, 101, 105, 136, 190
informaticien	Une entité active chargée de transformer les besoins métier en une application opérationnelle, 14, 19, 45, 54, 77, 97, 129
ingénierie des applications web	Est l'utilisation des principes scientifiques, d'ingénierie, de principes de gestion et les approches systématiques dans le but de réussir à développer, déployer et maintenir à haut niveau de qualité des systèmes et des applications basés sur le Web, 53
Ingénierie Dirigée par les Modèles	Est une approche d'ingénierie centrée sur le modèle. Elle offre un cadre méthodologique et technologique prometteur permettant d'unifier différentes façons de faire en un processus homogène et de favoriser l'étude des différents aspects du système, 4, 37, 38, 197
Intelectual Propertie	Est un composant dont l'utilisateur ne connaît que l'interface d'utilisation, l'implémentation interne reste la propriété du concepteur du composant, 128
IP	Intelectual Propertie (propriété intellectuelle), 128, 184
langage de transformation spécifique au domaine	Est une transformation décrite par un métamodèle spécifique au domaine métier, 45, 142
langage naturel	Est le langage parlé, par exemple le français ou l'anglais, 45
langage technologique	Est le langage utilsé pour implémenter une application, par exemple Python, Java, etc, 39
métier formel	Un modèle décrivant les besoins métier de l'application d'une manière en suivant un formalisme dédié. Dans ce modèle, les besoins sont complets, 39, 43

métier informel — Un modèle décrivant les besoins métier de l'application d'une manière informelle. Dans ce modèle, les besoins ne sont pas complets, 39, 43

MDA — Model Driven Architecture (architecture dirigée par les modèles) est une variante de l'IDM préconisant l'utilisation du lange de modélisation UML, 60

MOA — Modélisation Orientée Aspect (Aspect-Oriented Modeling), 141

modélisation des exigences — C'est la modélisation des besoins métier d'une application, 43, 53

modélisation orientée aspect — La modélisation par aspect est une forme de modélisation par composition de modèle visant à simplifier la construction des modèles, 141, 200

modèle — Une simplification du système réel permettant de répondre à toutes les questions sousjacentes à un point de vue donné du système, 37, 38

modèle technique — Est un modèle permettant de décrire l'application au niveau technique comme UML pour les approches orientées objet, 46

modèle technologique — Est un modèle permettant de décrire l'application au niveau technologique comme le Python ou le HTML, 46, 53

nœud de branchement exclusif à choix manuel — Est un nœud de contrôle permettant d'orienter le flux d'exécution d'un processus métier. Avec ce nœud le contrôle est fait par l'utilisateur exécutant le processus, 137–139, 191, 200

navigation fonctionnelle — Cette navigation permet d'effectuer un traitement sur les données de l'application web, 100, 132

navigation structurelle — Cette navigation permet de passer d'une page web à une autre sans conséquence sur les données de l'application. C'est donc une navigation à but informatif, 90, 100, 123, 132, 134

NF — Noyau Fonctionnel, 61, 62, 101, 136, 200

noyau fonctionnel — C'est le métier d'une application web (données, processus métier, services, etc), 73, 101, 133, 146, 190, 200

orchestration — C'est l'exécution des processus métier suivant un modèle d'exécution centré sur les activités (ou tâches). Dans ce cas, les activités sont considérées comme des entités explicites directement manipulables par les acteurs. Ces activités sont présentées aux acteurs sous forme d'une liste de tâches à réaliser séparément des données sur lesquelles elles s'appliquent (travailler sur les activités pour agir sur les données), 3, 4, 29, 30, 32, 68, 69, 77, 81, 98, 125, 200

plan documentaire — L'ensemble des informations et relations entre ces informations. C'est la vue informationnelle du système d'information, 62, 104, 146

point de vue de modélisation — Une vision particulière de l'entreprise qui met en lumière certains aspects, en augmentant leur niveau de détails, et rend transparent les autres, en diminuant leur niveau de détails, 17, 199

portail collaboratif — Est une application web permettant aux utilisateurs de travailler d'une manière collaborative sur des documents et selon des processus métier bien définis, 3, 146, 197

processus d'implémentation — Un processus instantané couvrant une fonctionnalité comme un enchaînement de formulaires. Il est assimilé à une fonctionnalité par exemple une fonctionnalité d'impression ou d'envoi de courrier électronique, 58, 63, 77, 79

processus de production de logiciels — Un enchainement d'activités, par exemple des transformations de modèles ou des générations de code, visant à transformer un ensemble de connaissances d'entrée en un ensemble de connaissances de sortie. Une connaissance peut être un modèle ou un code, 7, 40, 43, 45, 47, 49, 77, 145, 147, 154, 167, 171, 177, 198

processus de réalisation d'une action
: Est le processus décrivant les différentes étapes suivies par l'utilisateur et l'application afin de réaliser une action. Ce processus commence par le déclenchement d'une action jusqu'à l'affichage du contexte de post-étude de l'action, 133, 205

processus imbriqué
: Est un processus indépendant déclenché par une action d'un autre processus. Le processus imbriqué a un contexte d'exécution différent de celui du processus père, 114, 147, 177, 180, 190, 200

processus métier
: "*un ensemble de procédures et d'activités plus ou moins liées qui réalisent collectivement un objectif métier, en général au sein d'une structure organisationnelle définissant des rôles et des relations fonctionnelles.*" : Workflow Management Coalition, 3, 14, 15, 20, 27, 51, 56, 63, 77, 79, 85, 87, 88, 98, 114, 146, 172, 197

production
: Le passage d'une description technique des besoins métier à une description technologique, 40

rôle
: C'est la « fonction » professionnelle d'un acteur qui lui donne le droit d'accéder et/ou d'agir sur les instances des objets de collaboration, 18, 28, 87, 108, 146, 190, 199

racine de l'application
: C'est le point d'entrée d'une application web. Toutes les pages web d'une application sont accessibles à partir de la racine de l'application. Dans notre approche la racine de l'application est modélisée par le concept « Application », 87, 93

relation désignée
: Une relation entre une instance d'objet de collaboration et un processus métier. Dans cette relation l'instance est soit créée par le processus métier, soit choisi par l'utilisateur ou par le système pendant l'exécution, 108, 118

relation non désignée
: Une relation entre une instance d'objet de collaboration et un processus métier. Dans cette relation l'instance n'est pas connue d'avance et est quelconque (c'est-à-dire généralisée), 108, 118

remontée d'abstraction	La remontée d'abstraction consiste à simplifier le métamodèle en enlevant les détails techniques et technologiques et en s'approchant le plus possible du langage humain, en utilisant les termes et les concepts du domaine à modéliser, 76, 78, 82, 198
séparation des préoccupations	La séparation des préoccupations améliore la lisibilité des modèles et permet leur évolution en traitant chacune des préoccupations indépendamment les unes des autres, 76, 78, 82, 198
SI	Système d'Information, 22, 25
SIC	Système d'Information Collaboratif, 24, 25
site web "catalogue"	Site web à forte densité de données qui est caractérisé par le fait qu'il publie une masse importante de données, selon une structure hypertexte complexe, mais pauvre en services, 55, 65
site web de présence sur le Web	Un site web qui a comme objectif la mise en ligne d'informations, à des fins publicitaires ou éducatives par exemple, 55, 65
site web orienté services	Site web destiné à offrir des services spécifiques comme les applications de messagerie électronique, 56
SIW	Système d'Information basé sur le Web, 54, 56, 63
sous-processus	Est un processus englobé par un processus père. Le sous-processus partage le même contexte d'exécution que le processus père. Le sous-processus est associé à une action du processus père. Le déclenchement de l'action correspond au déclenchement du sous-processus, et la fin du sous-processus correspond à la fin de l'action, 114
SPPI	Socioculturelles, et différences Psychologiques, Physiologiques et Idéologiques, 72, 75, 76, 82, 198

syntaxe abstraite	La syntaxe abstraite d'un métamodèle est ce qui correspond aux instances des métaéléments (concepts et relations) du métamodèle. Cette syntaxe est unique, 74, 82, 198
syntaxe concrète	La syntaxe concrète d'un métamodèle est une représentation graphique ou textuelle des concepts du métamodèle. Un même métamodèle peut avoir plusieurs syntaxes concrètes, 74, 76, 198
systéme physique	Le système constitué du réel et virtuel d'un système donné, 22, 198
système	Un ensemble d'éléments qui interagissent entre eux selon un enchaînement d'actions organisées et en vue d'atteindre un but bien défini, 13, 20
système d'information	Le système constitué d'information sur le réel et virtuel d'un système donné, 3, 16, 19–23, 25, 32, 34, 101, 126, 146, 197
système d'information basé sur le Web	Application web qui combine la mise à disposition et la gestion de données complexes avec des services interactifs orchestrés par des processus métier complexes, 54, 56
système d'information collaboratif	Un système d'information qui inclut les différentes parties publiques des systèmes d'information des entreprises participantes à une collaboration donnée, 14, 20, 24, 32, 34
système informatique	Un système qui fait partie du système physique et est constitué des moyens informatiques (Application, bases de données, ordinateur, etc.), 21, 25
système informatique collaboratif	Un système informatique qui inclut les différentes parties publiques des systèmes informatique des entreprises participantes à une collaboration donnée en plus de la partie spécifique à la collaboration, 14, 25, 32, 34

système opérant	Selon l'approche fonctionnelle, le système opérant permet de transformer des entrées (données à l'état primaire) en sorties (produits ou services transformés pour une finalité identifiée), 20, 21
virtualisation	Le passage d'une description métier formelle des besoins métier à une description technique, 40
vue	Une vue est un ensemble de structures de données visuelles (texte, image, formulaire, etc.). Une page web dans nos portails collaboratifs est composée de plusieurs vues, 99, 123, 133, 147, 199
vue des ressources	La vue des ressources fournit une représentation de l'ensemble des moyens nécessaires pour mettre en oeuvre les activités de l'entreprise, 17, 103, 146, 172, 190, 199
vue fonctionnelle	La vue fonctionnelle fournit une représentation des processus métier de l'entreprise, 17, 103, 146, 172, 190, 199
vue informationnelle	La vue informationnelle fournit une représentation structurée d'un ensemble d'informations de l'entreprise et des relations de dépendance entre ces informations, 17, 103, 146, 172, 190, 199
vue organisationnelle	La vue de l'organisation fournit une représentation de l'organisation structurelle de l'entreprise, 17
WCMS	Système de gestion de contenu basé sur le web, 33
WfMC	Workflow Management Coalition, 27, 29
WMS	Système de gestion de workflow, 28, 31
Workflow	Un workflow présente une vision automatisable d'un processus métier, 28, 33
XSL	eXtensible Stylesheet Language, 68

BIBLIOGRAPHIE

[1] Roberto Acerbis, Aldo Bongio, Marco Brambilla, Stefano Butti, Stefano Ceri, and Piero Fraternali. Web applications design and development with webml and webratio 5.0. 11 :392–411, 2008.

[2] Rui Silva Carlos Martins Alberto Rodrigues da Silva, Joao Saraiva. Xis – uml profile for extreme modeling interactive systems, 2004.

[3] Schmid Hans Albrecht and Rossi Gustavo. Modeling and designing processes in e-commerce applications. *Internet Computing IEEE*, 8(1) :19–27, 2004.

[4] Ravi Sethi Alfred Aho, Monica Lam and Jeffrey Ullman. *Compilateurs : principes, techniques et outils*. Hermes Science publications – Lavoisier, 2 edition, nov 2007.

[5] Timothy C. Lethbridge Ali Fatolahi, Stéphane S. Somé. Automated generation of abstract web applications using qvt relations, 2009.

[6] Thomas Allweyer. *BPMN 2.0*. BoD, feb 2010.

[7] Victor Anaya, Giuseppe Berio, Mounira Harzallah, Patrick Heymans, Raimundas Matulevicius, Andreas L. Opdahl, Hervé Panetto, and Maria Jose Verdecho. The unified enterprise modelling language–overview and further work. *Computers in Industry*, 61(2) :99–111, feb 2010.

[8] Colette Rolland André Flory. Nouvelles perspectives des systèmes d'information. *Congrés 90 de l'Association informatique des organisations et systèmes d'information et de décision, Paris*, pages 3–40, 1990.

[9] Kleppe Anneke. Mcc : A model transformation environment. In Warmer Jos Rensink Arend, editor, *Model Driven Architecture – Foundations and Applications*, number 4066 in Lecture Notes in Computer Science, pages 173–187. Springer Berlin Heidelberg, jan 2006. ISBN 978-3-540-35909-8, 978-3-540-35910-4.

[10] AspectJ. Aspectj, 2013. URL http://eclipse.org/aspectj/.

[11] ATHENA. Athena. http ://www.athena-ip.org/. URL http://www.athena-ip.org/.

[12] M. Franck BARBIER. Enterprise model-driven development with blu age.

[13] Luciano Baresi, Sebastiano Colazzo, Luca Mainetti, and Sandro Morasca. W2000 : A modelling notation for complex web applications. pages 335–364. 2006.

[14] Koch Nora Baumeister Hubert, Knapp Alexander and Zhang Gefei. Modelling adaptivity with aspects. In *Web Engineering*, pages 406–416. Springer, 2005.

[15] P. Bernus and L. Nemes. A framework to define a generic enterprise reference architecture and methodology. *Computer-integrated manufacturing systems*, 9(3) :179–191. ISSN 0951-5240.

[16] Michael Bieber. Hypertext and web engineering, 1998.

[17] Boiko Bob. *ontent Management Bible*. John Wiley and Sons, nov 2005.

[18] BonitaSoft. Bonitasoft, 2013. URL http://fr.bonitasoft.com/.

[19] Am è lie Boucher. *Ergonomie web : Pour des sites web efficaces*. Eyrolles, 3e é dition edition, oct 2011.

[20] Marco Brambilla, Sara Comai, Piero Fraternali, and Maristella Matera. Designing web applications with webml and webratio. In Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina, editors, *Web Engineering : Modelling and Implementing Web Applications*, Human-Computer Interaction Series, pages 221–261. Springer London, 2008.

[21] Marco Brambilla, Stefano Butti, and Piero Fraternali. Webratio bpm : a tool for designing and deploying business processes on the web. In *Proceedings of the 10th international conference on Web engineering*, ICWE'10, pages 415–429. Springer-Verlag, 2010.

[22] Arnaud Brossard, Mourad Abed, and Christophe Kolski. Modélisation conceptuelle des ihm. une approche globale s'appuyant sur les processus métier. *Ingénierie des systèmes d'information*, 12(5) :69–108, dec 2007.

[23] Cristina Cachero, Jaime Gomez, and Oscar Pastor. Object-oriented conceptual modeling of web application interfaces : the oo-hmethod abstract presentation model. In *Proceedings of the First International Conference on Electronic Commerce and Web Technologies*, EC-WEB '00, pages 206–215. Springer-Verlag, 2000.

[24] Francisco José Cabral Cardoso. A mde approach for the development of cms-based web applications, 2009.

[25] Cutter Consortium. Poor project management number-one problem of outsourced e-projects, 2000. URL http://www.cutter.com/research/2000/crb001107.html.

[26] Gerardo Canfora Damiano Distante, Gustavo Rossi. Modeling business processes in web applications :an analysis framework, March 2007.

[27] Franck Darras. Proposition d'un cadre de référence pour la conception et l'exploitation d'un progiciel de gestion intégré., oct 2004.

[28] Marcelo Paternostro Dave Steinberg, Frank Budinsky and Ed Merks. *EMF : Eclipse Modeling Framework*. 2 edition, dec 2008.

[29] Richard Hull David Cohn. Business artifacts : A data-centric approach to modeling business operations and processes. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2009.

[30] Lange D.B. An object-oriented design method for hypermedia information systems. In *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, 1994*, volume 3, pages 366 –375, jan 1994.

[31] Django. Django, 2013. URL https://www.djangoproject.com/.

[32] Guy Doumeingts. *Méthode GRAI : méthode de conception des systèmes en productique*. Thèse, 1984.

[33] M. Lewkowicz E. Soulier. Simulation des pratiques collaboratives pour la concpetion des si basés sur les processus métiers, revue ingénierie des systèmes d'information (isi). 11(3), 2006.

[34] Eclipse. Acceleo. http://wiki.eclipse.org/Acceleo, .

[35] Eclipse. Atl, . URL http://www.eclipse.org/atl/.

[36] Eclipse. Eclipse, . URL http://www.eclipse.org/.

[37] Eclipse. Mwe. http://wiki.eclipse.org/MWE/oaw4.3_doc, .

[38] Eclipse. Qvto, . URL https://projects.eclipse.org/projects/modeling.mmt.qvt-oml.

[39] Christoph Eickhoff, Nina Geiger, Marcel Hahn, and Albert Zundorf. Developing enterprise web applications using the story driven modeling approach. In *Current Trends in Web Engineering*, volume 7059, pages 196–210. Springer Berlin Heidelberg, 2012.

[40] Lauri Eloranta, Eero Kallio, and Ilkka Terho. A notation evaluation of bpmn and uml activity diagrams. pages 1–45, 2006.

[41] M. Didonet Del Fabro and F. Jouault. Model transformation and weaving in the amma platform. *Workshop on Generative and Transformational Techniques*, pages 71–77, 2005.

[42] Chen Fang. *Designing Human Interface in Speech Technology*. Springer, 2006.

[43] Roozbeh Farahbod, Uwe Glasser, and Mona Vajihollahi. Specification and validation of the business process execution language for web services. *Abstract State Machines 2004 Advances in Theory and Practice*, (1) :78–94, 2004.

[44] J.-M. Favre. *Concepts fondamentaux de l'IDM. De l'ancienne egypte à l'ingénierie des langages*. 2èmes Journées sur l'Ingénierie Dirigée par les Modèles (IDM'06). Lille, France, 2006.

[45] Layna Fischer. *2010 BPM and Workflow Handbook, Spotlight on Business Intelligence*. Future Strategies Inc, jun 2010.

[46] C. Floyd. A comparative evaluation of system development methods, 1986.

[47] Joan Fons, Pedro Valderas, Marta Ruiz, Gonzalo Rojas, and Oscar Pastor. Oows : A method to develop web applications from web-oriented conceptual models. 2003.

[48] PIERO FRATERNALI. *Tools and Approaches for Developing Data-Intensive WebApplications : A Survey*, volume 31. ACM Computing Surveys, September 1999.

[49] Vallecillo-Moreno A Fuentes-Fernández L. An introduction to uml profiles. *UPGRADE, European Journal for the Informatics Professional*, 5(2) :5–13, apr 2004.

[50] P. Atzeni V. Crescenzi G. Mecca, P. Merialdo. *The ARANEUS Guide to Web-Site Development*. ARANEUS Project Working Report, 1999.

[51] Claude Godart and Olivier Perrin. *Les processus métiers : Concepts, modèles et systèmes*. Hermes Science Publications, sep 2009.

[52] B. Griffiths G. Lockyer M. Oates and B. Hebbron. Empirical methodologies for web engineering, proceedings, 2004.

[53] Kolovos D. S. Paige R. F. dos Santos O. M. Guerra E., de Lara J. Engineering model transformations with transml. *Software and Systems Modeling*, page 1–23, 2011. URL http://www.springerlink.com/index/88W21765P1277683.pdf.

[54] Nguyen Viet Hoa. Automatisation de l'enchaînement des tâches exécutées sur des modèles. Rapport de fin d'étude, 2009.

[55] IBM. Ibm - websphere business modeler advanced (wbm), 2013. URL https://developers.google.com/web-toolkit/.

[56] IDEF. Integration definition for function modeling (idefo). Technical report, 1993.

[57] ISO. Enterprise integration - framework for enterprise modelling.

[58] ISO. Iso 9241, 2013. URL http://www.iso.org/iso/fr/home/search.htm?qt=ISO+9241&sort=rel&type=simple&published=on.

[59] H. Pingaud J. Touzi, F. Bénaben. Collaborative information system design : From process model to information system model, May 17-19 2006.

[60] Sonia JAMAL. Environnement de procédé extensible pour l'orchestration application aux services web, 2005.

[61] F. Sánchez J.C. Preciado, M. Linaje. Necessity of methodologies to model rich internet applications, 2005.

[62] Bézivin Jean. In search of a basic principle for model driven engineering. *UPGRADE – The European Journal for the Informatics Professional*, 5(2) :21–24, 2004.

[63] Jacky Estublier Jean-Marie Favre and Mireille Blay-Fornarino. *L'ingénierie dirigée par les modèles : au-delà du MDA*. Hermes Science publications – Lavoisier, feb 2006.

[64] Jean-Marc Jézéquel. Le génie logiciel et l'idm : une approche unificatrice par les modèles, 2006.

[65] Jean-Marc Jézéquel. Model driven design and aspect weaving, 2008.

[66] Jean-Marc Jézéquel, Benoît Combemale, and Didier Vojtisek. *Ingénierie Dirigée par les Modèles : des concepts a la pratique*. Ellipses Marketing, February 2012.

[67] Touzi Jihed. Aide à la conception de système d'information collaboratif , support de l'interopérabilité des entreprises, nov 2007.

[68] Conallen Jim. *Building Web Applications with UML*. Addison Wesley, 2 edition, oct 2002.

[69] Alberto Rodrigues da Silva João de Sousa Saraiva. Cms-based web-application development using model-driven languages. In *Proceedings of the 2009 Fourth International Conference on Software Engineering Advances*, ICSEA '09, pages 21–26. IEEE Computer Society, 2009.

[70] Réda KADRI. *Une approche pour la Modélisation d'Applications Web a base de Composants Logiciels*. PhD thesis, jan 2009.

[71] Sam Knox, Jon Stahl, Martin Aspeli, David Convent, Darci Hanning, Ricardo Newbery, John DeStefano, Clayton Parker, Alex Clark, and Veda Williams. *Practical Plone 3 : A Beginner's Guide to Building Powerful Websites*. Packt Publishing, February 2009.

[72] Ryan K. L. Ko. A computer scientist's introductory guide to business process management (bpm). *ACM Crossroads*, 15(4), 2009.

[73] Birgit Korherr. Business process modelling -languages, goals, and variabilities, 2008.

[74] K Kosanke, F Vernadat, and M Zelm. Cimosa :enterprise engineering and integration. *Computers in Industry*, 40(2-3) :83–97, November 1999.

[75] Garcia Alessandro-Chavez Christina Krechetov Ivan, Tekinerdogan Bedir and AOSD" volume="6" year="2006" organization="Citeseer" Kulesza Uirá", booktitle="8th Workshop on Aspect-Oriented Modelling (AOM. 06). Towards an integrated aspect-oriented modeling approach for software architecture design.

[76] Christian Kroiss and Nora Koch. Uwe metamodel and profile . user guide and reference. Technical report, feb 2008.

[77] Christian Kroiss, Nora Koch, and Alexander Knapp. Uwe4jsf : A model-driven generation approach for web applications. In *Proceedings of the 9th International Conference on Web Engineering*, ICWE '9, pages 493–496. Springer-Verlag, 2009.

[78] Philippe Kruchten. *The Rational Unified Process : An Introduction*. Addison-Wesley Professional, 2 edition, mar 2000.

[79] Kosanke Kurt. Iso standards for interoperability : a comparison. In *Interoperability of Enterprise Software and Applications*, pages 55–64. Springer, 2006.

[80] Olsina L. Building a web-based information system applying the hypermedia flexible process modeling strategy, 1998.

[81] Heeseok Lee, Choongseok Lee, and Cheonsoo Yoo. *A Scenario-Based Object-Oriented Methodology for Developing Hypermedia Information Systems*, volume 2 of *HICSS '98*. IEEE Computer Society, 1998.

[82] Thomas Lèvêque. Définition et contrôle des politiques d'évolution dans les projets logiciels, 2010.

[83] J. Bézivin M. Didonet Del Fabro and P. Valduriez. Weaving models with the eclipse amw plugin, 2006.

[84] NORA KOCH M. José ESCALONA. Requirements engineering for web applications – a comparative study. *Web Engineering*, 2 (3), 2004.

[85] O. Macek and K. Richta. The bpm to uml activity diagram transformation using xslt. pages 119–129, 2009.

[86] STEFANO CERI MARCO BRAMBILLA and PIERO FRATERNALI. Process modeling in web applications. *ACM Transactions on Software Engineering and Methodology*, 15(4) :360–409, October 2006.

[87] John McDermid and Knut Ripken. Life cycle support in the ada environment. III(1) :57 – 62, July-August 1983.

[88] Vasundhara Fegade Md. Sadique Shaikh. Modeling essentials of content management system (cms) for web-based mis application. *International Journal of Engineering and Technology*, 2(3), March 2012.

[89] Thomas Leveque Mehrdad Saadatmand. Modeling security aspects in distributed real-time component-based embedded systems. *ITNG*, 2012.

[90] Tomás Isakowitz Michael Bieber. Designing hypermedia applications. *COMMUNICATIONS OF THE ACM*, 38(8), 1995.

[91] Pierre-Alain Millet. *Une étude de l'intégration organisationnelle et informationnelle : Application aux systèmes d'informations de type ERP*. PhD thesis, L'institut national des sciences appliquées de Lyon, oct 2008.

[92] F.M. Martin-J. Nieto A. Llergo F. Pérez M.J. Escalona, C.L. Parra. A practical example for model-driven web requirements, 2009.

[93] Matthew Moodie. *Pro Apache Ant*. Apress, 1 edition, jan 2012. ISBN 1430243112.

[94] Chantal Morley, Jean Hugues, and Bernard Leblanc. *UML2 pour l'analyse d'un système d'information : Le cahier des charges du maître d'ouvrage*. Dunod, 3e édition edition, jan 2006.

[95] Chantal Morley, Jean Hugues, Bernard Leblanc, and Olivier Hugues. *Processus métiers et systèmes d'informations : Evaluation, modélisation, mise en oeuvre*. Dunod, 2e édition edition, nov 2007.

[96] Koch Nora. Hypermedia systems development based on the unified process. *Ludwig-Maximilians-University Munich, Institute of Computer Science*, 2000.

[97] Thomas Ledoux Noury M. N. Bouraqadi-Saâdani. Le point sur la programmation par aspects. *Technique et science informatiques*, 20(4) :505–528, 2001.

[98] C.J. Leune O.M.F. De Troyer. Wsdm : a user centered design method for web sites.

[99] OMG. Mof model to text transformation language (mofm2t), 1.0, . URL http://www.omg.org/spec/MOFM2T/1.0/.

[100] OMG. Omg, . URL http://www.omg.org/.

[101] OMG. Business process definition metamodel (bpdm), version 1.0, . URL http://www.omg.org/spec/BPDM/1.0/.

[102] OMG. Omg's metaobject facility, . URL http://www.omg.org/mof/.

[103] OMG. Object constraint language omg available specification version 2.0, . URL http://www.omg.org/spec/OCL/2.0.

[104] OMG. Documents associated with unified modeling language (uml), v2.4.1, . URL http://www.omg.org/spec/UML/2.4.1/.

[105] OMG. Documents associated with meta object facility (mof) 2.0 query/view/transformation, v1.1, . URL http://www.omg.org/spec/QVT/1.1/.

[106] OMG. Spem2.0. http://www.omg.org/spec/SPEM, .

[107] OMG. Business process model and notation (bpmn) ftf beta 1 for version 2.0, 2011. URL http://www.omg.org/spec/BPMN/2.0/.

[108] OMG. Interaction flow modeling language (ifml) version 1.0, 2013. URL www.omg.org/spec/IFML/1.0.

[109] Wong P. and Gibbons J. A process semantics for bpmn. *Formal Methods and Software Engineering*, page 355–374, 2008.

[110] Michel Pelletier. *Plone Live*. LL, 2005.

[111] Magnus Penker and Hans-Erik Eriksson. *Business Modeling With UML : Business Patterns at Work*. Wiley, 1 edition, 2000.

[112] Muller Pierre-Alain, Studer Philippe, Fondement Frédéric, and Bézivin Jean. Platform independent web application modeling and development with netsilon. *Software & Systems Modeling*, 4 (4) :424–442, 2005.

[113] López-romero F. Bautist J. Vallecillo A. Rivera J. E., Ruiz-gonzález D. Orchestrating atl model transformations. In *In Jouault [7]*.

[114] Esteban Robles Luna, Julián Grigera, and Gustavo Rossi. Bridging test and model-driven approaches in web engineering. In Martin Gaedke, Michael Grossniklaus, and Oscar Diaz, editors, *Web Engineering*, volume 5648 of *Lecture Notes in Computer Science*, pages 136–150. Springer Berlin / Heidelberg, 2009.

[115] Esteban Robles Luna, Gustavo Rossi, and Irene Garrigos. Webspec : a visual language for specifying interaction and navigation requirements in web applications. *Requirements Engineering*, 16 :297–321, 2011.

[116] Hennicker Rolf and Koch Nora. Modeling the user interface of web applications with uml. *pUML*, 7 :158–172, 2001.

[117] Pascal Roques. *UML 2 : Modéliser une application web*. Eyrolles, 3e édition edition, apr 2007.

[118] Joël Rosnay. *Le macroscope : vers une vision globale*. Éditions du Seuil, 1975.

[119] A. Ginige S. Murugesan. Web engineering : Introduction and perspectives, 2005.

[120] SADT. Structured analysis design technic (sadt). URL http://web.univ-pau.fr/nancy/sadt/.

[121] Vinh Le Thai-Bernard Coulette Samba Diaw, Redouane Lbath. Spem4mde : a metamodel for mde software processes modeling and enactment. *3rd Workshop on Model-Driven Tool & Process Integration - Associated to EC-MFA*, 2010.

[122] Nora Koch Santiago Meliá, Jaime Gómez. Improving web design methods with architecture modeling, 2005.

[123] Jorge L. C. Sanz. Entity-centric operations modeling for business process management a multidisciplinary review of the state-of-the-art. *Proceedings of The 6th IEEE International Symposium on Service Oriented System Engineering*, 2011.

[124] Schwinger Wieland Kapsammer Elisabeth Schauerhuber Andrea, Wimmer Manuel and Retschitzegger Werner. Aspect-oriented modeling of ubiquitous web applications : The aspectwebml approach. In *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, pages 569–576. IEEE, 2007.

[125] August-Wilhelm Scheer. *ARIS : Des processus de gestion au système intégré d'applications*. 2008-07-03.

[126] Daniel Schwabe, Rita de Almeida Pontes, and Isbela Moura. Oohdm-web : an environment for implementation of hypermedia applications in the www. *SIGWEB Newsl.*, 8(2) :18–34, June 1999.

[127] Michel Serres. Les nouvelles technologies : révolution culturelle et cognitive, 2007. URL https://interstices.info/jcms/c_33030/les-nouvelles-technologies-revolution-culturelle-et-cognitive.

[128] Berners-Lee Tim Shadbolt Nigel and Hall Wendy. The semantic web revisited. *IEEE Intelligent Systems*, 21(3) :96–101, may 2006. ISSN 1541-1672.

[129] Bruce Silver. *BPMN Method and Style : A levels-based methodology for BPM process modeling and improvement using BPMN 2.0*. Cody-Cassidy Press, jun 2009.

[130] Jean-Sébastien Sottet. *Mega-IHM : Malléabilité des Interfaces Homme-Machine Dirigée par les Modèles*. PhD thesis, JOSEPH FOURIER - GRENOBLE 1, oct 2008.

[131] Jonathan Sprinkle. Analysis of a metamodel to estimate complexity of using a domain-specific language. 2010.

[132] Piero Fraternali Stefano Ceri, Marco Brambilla. The history of webml lessons learned from 10 years of model-driven development of web applications, 2009.

[133] Struts. Struts, 2013. URL http://struts.apache.org/.

[134] Noi Sukaviriya, Vibha Sinha, Thejaswini Ramachandra, Senthil Mani, and Markus Stolze. User-centered design and business process modeling : cross road in rapid prototyping tools. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction*, INTERACT'07, pages 165–178. Springer-Verlag, 2007.

[135] Symphony. Symphony, 2013. URL http://symfony.com/.

[136] LIANG E. TAKAHASHI K. *Analysis and design of Web-based informations systems*. 1997.

[137] NGUYEN Thi Thanh Tam. Codèle : Une approche de composition de modèles pour la construction de systèmes à grande échelle, 2008.

[138] Hubert Tardieu, Arnold Rochfeld, and René Colletti. *La méthode Merise. Principes et outils*. Editions d'Organisation, 2000. ISBN 2708124730.

[139] Dominique C. Cutts. Thomas A. Powell, David L. Jones. Web site engineering : beyond web page design, 1998.

[140] P. Balasubramanian Tomás Isakowitz, Edward A. Stohr. Rmm : A methodology for structured hypermedia design, 1995.

[141] Google Web Toolkit. Google web toolkit, 2013. URL https://developers.google.com/web-toolkit/.

[142] Victoria Torres and Vicente Pelechano. Building business process driven web applications. In *Proceedings of the 4th international conference on Business Process Management*, BPM'06, pages 322–337. Springer-Verlag, 2006.

[143] Christophe Torset. La réflexion stratégique, objet et outil de recherche pour le management stratégique ? In *Actes de la XIVème conférence internationale de management stratégique*, 2005.

[144] David Travis. New standards in usability. 2009. URL http://www.userfocus.co.uk/articles/ISO9241_update.html.

[145] Baelen Stefan Van Joosen Wouter Berbers E. Vanhooff Bert, Ayed Dhouha. Uniti : A unified transformation infrastructure. In *In ACM/IEEE 10th International Conference on Model-Driven Engineering Languages and Systems (MoDELS 2007)*, 2007.

[146] Marlène VILLANOVA-OLIVER. Adaptabilit|'e dans les systémes d'information sur le web : Modélisation et mise en œuvre de l'accés progressif, dec 2002.

[147] Lam V. S. W. A precise execution semantics for bpmn. *IAENG International Journal of Computer Science*, 39.

[148] WFMC. rapport, 2008. URL www.wfmc.org.

[149] wikipedia. Qqoqccp. URL http://fr.wikipedia.org/wiki/QQOQCCP.

[150] El Kaim William, Studer Philippe, and Muller Pierre-Alain. Model driven architecture for agile web information system engineering. In *Object-Oriented Information Systems*, pages 299–303. Springer, 2003.

[151] Wen Xinxiu and Yu Huiqun. An aspect-oriented modeling approach. In *Software Engineering (WCSE), 2010 Second World Congress on*, volume 1, pages 307–310. IEEE, 2010.

[152] Sira Yongchareon, Chengfei Liu, Xiaohui Zhao, and Jiajie Xu. An artifact-centric approach to generating web-based business process driven user interfaces. In *Proceedings of the 11th international conference on Web information systems engineering*, WISE'10, pages 419–427. Springer-Verlag, 2010.

[153] Chen Yuejuan Zhang Jingjun and Liu Guangyuan. Modeling aspect-oriented programming with uml profile. In *Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on*, volume 2, pages 242–245. IEEE, 2009.

MODÉLISATION CENTRÉE SUR LES PROCESSUS MÉTIER POUR LA GÉNÉRATION COMPLÈTE DE PORTAILS COLLABORATIFS

Les entreprises collaborent pour saisir des opportunités, échanger des documents et ressources, cela en suivant des processus métier pouvant évoluer. Les portails collaboratifs sont une solution orientée web à ce besoin de collaboration. Cependant, la conception et la maintenance d'un portail collaboratif métier n'est pas trivial et reste peu accessible aux acteurs de l'entreprise. Cela a comme conséquence la difficulté de maintenir et faire évoluer le portail collaboratif sans que cela ne soit trop coûteux en temps et financièrement. Afin de répondre à cette problématique, une solution consiste à capter les besoins métier de la collaboration dans un modèle, puis générer automatiquement le portail collaboratif correspondant. Le modèle, dans ce cas, doit être accessible aux acteurs métier et expressif décrivant ainsi les aspects les plus complexes d'une collaboration. C'est dans ce contexte que se situent nos travaux. À défaut d'avoir une solution toute faite, nous avons mis en place une approche de conception de portail collaboratif fondée sur l'Ingénierie Dirigée par les Modèles. Pour la description de nos portails, nous avons choisi de privilégier la modélisation des entreprises centrée sur les processus métier comme point de départ. Notre solution repose sur notre métamodèle MACoP (Modeling and Analysis of Collaborative Portal). Dans ce métamodèle nous avons fait cohabiter l'accessibilité et l'expressivité. Cela en proposant de nouveaux concepts permettant ainsi la génération complète des portails collaboratifs. Le métamodèle MACoP est accompagné d'une chaîne de transformations permettant de passer directement d'un modèle MACoP au code Python du portail collaboratif.

Mots clés : Portail collaboratif, Application web, Processus métier, Système d'information, Modélisation des entreprises, Ingénierie Dirigée par les Modèles (IDM), Transformation de modèles, Génération de code.

BUSINESS PROCESS-CENTERED MODELING FOR THE COMPLETE GENERATION OF COLLABORATIVE PORTALS

Companies collaborate to seize opportunities as well as exchange documents and other types of resources. This is achieved by following business processes that are subject to evolution. Collaborative portals are web oriented solutions aimed at this need of collaboration. However, the development and maintenance of a collaborative portal is non-trivial and remains hardly accessible for many companies. As a consequence, the challenge is controlling the costs of maintenance and implementing new features. To circumvent these issues, it is possible to collect business requirements of the collaboration in a model and then generate automatically the corresponding collaborative portal. The model, in this case, must be accessible by the business actors and must express even the most complex aspects of the collaboration needs. This is the context of our work. Having not found a solution which corresponded to our requirements, we have developed a design approach of collaborative portals founded on the Model Driven Engineering. As for the description of our portals, we have chosen to focus on the business modeling based on the business processes like starting point. Our solution relies on our meta-model MACoP (Modeling and Analysis of Collaborative Portal). In this meta-model we have joined together accessibility and the expressivity, by proposing new concepts allowing the complete generation of collaborative portals. The meta-model MACoP is accompanied by a transformations chain that makes it possible to pass directly from a MACoP model to the Python code of the collaborative portal.

Keywords : Collaborative portal, web application, business process, information system, business modeling, Model Driven Engineering (MDE), Models transformation, code generation.